Residual TPP: A Unified Lightweight Approach for Event Stream Data Analysis

Ruoxin Yuan [1] **Guanhua Fang** [1]

Abstract

This work introduces Residual TPP, a novel, unified, and lightweight approach for analyzing event stream data. It leverages the strengths of both simple statistical TPPs and expressive neural TPPs to achieve superior performance. Specifically, we propose the Residual Events Decomposition (RED) technique in temporal point processes, which defines a weight function to quantify how well the intensity function captures the event characteristics. The RED serves as a flexible, plug-and-play module that can be integrated with any TPP model in a wide range of tasks. It enables the identification of events for which the intensity function provides a poor fit, referred to as residual events. By combining RED with a Hawkes process, we capture the self-exciting nature of the data and identify residual events. Then an arbitrary neural TPP is employed to take care of residual events. Extensive experimental results demonstrate that Residual TPP consistently achieves state-of-the-art goodness-of-fit and prediction performance in multiple domains and offers significant computational advantages as well.

1. Introduction

Event stream data, characterized by sequences of events occurring over time, plays a crucial role across various domains, including financial transactions (Bacry et al., 2015), neuroscience (Williams et al., 2020), social media (Farajtabar et al., 2017), and natural disaster monitoring (Fox et al., 2016). In these fields, events are typically discrete, irregularly spaced, and may exhibit complex temporal dependencies, distinguishing them from conventional time series data. Consequently, many studies seek to understand and model the underlying dynamics of event streams in order to address tasks such as predicting future events (Du et al., 2016; Biloš et al., 2019; Shchur et al., 2020), detecting anomalies in event sequences (Liu & Hauskrecht, 2021a; Shchur et al., 2021), and performing causal inference (Xu et al., 2016; Zhang et al., 2020b; 2022).

Temporal Point Processes (TPPs) (Daley & Vere-Jones, 2003) are widely used to model the temporal structure and dependencies in event stream data. A TPP defines the probability of events occurring at a specific time, conditioned on the past event history, through an intensity function. Traditional statistical TPPs, such as the Possion process (Kingman, 1992) and the Hawkes process (Hawkes, 1971), exhibit favorable statistical properties with fixed parametric intensity functions. While these models are computationally simple and have been extensively applied in finance (Hasbrouck, 1991), seismology (Ogata, 1988), and other domains for decades, the strong parametric assumptions constrain their ability to capture the complexity of real-world event dynamics. To address the limitations and enhance the expressiveness of classical TPPs, numerous studies have leveraged the power of neural networks to construct neural TPPs. Most neural TPPs parameterize the intensity function using recurrent neural networks (Du et al., 2016; Mei & Eisner, 2017) and attention mechanisms (Zuo et al., 2020; Zhang et al., 2020a), which enable the capture of complex dependencies. Furthermore, research has explored innovative extensions of TPPs, including their integration into meta-learning frameworks (Bae et al., 2023) and modeling the intensity process as the solution to a neural jump-diffusion stochastic differential equation (Zhang et al., 2024).

Despite these efforts, several limitations remain in the analysis of event stream data. (i) RNN-based TPPs struggle to effectively capture long-term dependencies while transformer-based TPPs suffer from high computational costs, particularly when processing long sequences. (ii) There is a lack of focus on developing lightweight models. Existing models tend to prioritize accuracy, often at the expense of computational efficiency, limiting their scalability and practical generalization ability. (iii) Unlike in time series analysis, where techniques like *Seasonal-Trend Decomposition (STD)* have been successfully applied to decompose data into trend, seasonal, and residual components, no comparable decomposition methods have been proposed for TPPs to better capture the temporal structure of event streams and simplify the computational process.

[1]School of Management, Fudan University, Shanghai, China. Correspondence to: Guanhua Fang <fanggh@fudan.edu.cn>.

Proceedings of the 42^{nd} International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025.

In this paper, we pioneer the exploration of leveraging the statistical properties of event stream data, such as periodicity and self-excitation, to decompose the data and construct a lightweight model. Specifically, we introduce **Residual TPP**, a unified approach that combines the simplicity of traditional statistical TPPs with the expressiveness of neural TPPs, achieving superior performance. Technically, we propose the **Residual Events Decomposition** (RED) technique, which utilizes a weight function to separate residuals in temporal point processes. In Residual TPP, we first integrate RED with a Hawkes process to capture event characteristics and identify residual events. Subsequently, an arbitrary neural TPP is employed to handle the residual events, thereby reducing the computational burden of the neural TPP. Residual TPP is conceptually simple, computationally efficient, and demonstrates substantial improvements in goodness-of-fit and prediction accuracy across multiple domains.

In summary, the contributions of this paper are as follows:

- Due to the distinguished nature of event stream data (i.e. discrete event types and irregular event times), the STD methods deployed in time series data cannot be applied here. Therefore, we introduce a novel RED technique for TPPs, which uses a proper weight function to evaluate the performance of the intensity function and identify residual events. RED is the first TPP decomposition method and serves as a flexible, plug-and-play module that can be integrated with any TPP model to enhance its performance.
- Building on RED, we propose the Residual TPP, a unified approach that integrates both statistical TPPs and neural TPPs, demonstrating robust generalization capabilities.
- The extensive numerical results show that the proposed Residual TPP not only achieves consistent state-of-the-art performances across multiple domains but is also more computationally efficient than existing methods. Therefore, our new model is lightweight, reducing the demand for computational resources.

Notation. In this paper, we denote the total event dataset, the collection of event sequences, by $\mathbf{S} = \{S_n\}_{n=1}^N$ and the residual event dataset by $\mathbf{S}'_w = \{S'_{n,w}\}_{n=1}^N$, where N is the total number of sequences. We use $|\mathbf{S}|$ to denote the total number of events in all sequences. Model parameters for the Hawkes process and neural TPPs are represented by $\boldsymbol{\theta} = \{\theta_k\}_{k=1}^K$ and $\boldsymbol{\psi} = \{\psi_k\}_{k=1}^K$, respectively. The subscripts n, i, and k refer to the event sequence index, event number index, and event type, respectively.

2. Background

This section provides a brief introduction to the definition of temporal point processes, their statistical modeling methods and neural network-based modeling approaches for temporal point processes.

2.1. Temporal Point Processes

A temporal point process (TPP) is a type of stochastic process used to model the occurrence of events over time. Specifically, a temporal point process is a sequence of random variables denoted as $\{t_i\}_{i=1}^I$, where t_i represents the time of occurrence of the i-th event, with $0 < t_1 < \cdots < t_I \leq T$. A marked temporal point process (marked TPP) extends the traditional temporal point process by not only modeling the times at which events occur, but also modeling each event type with a discrete variable $k_i \in \{1, \cdots, K\}$. Formally, a marked temporal point process is a sequence denoted as $S = \{(t_1, k_1), \ldots, (t_I, k_I)\} = \{(t_i, k_i)\}_{i=1}^I$. In this paper, we primarily focus on marked TPP, as they represent a more general and widely applicable class of data.

A typical way to characterize a TPP is through its *intensity function*, denoted as $\lambda_k(t|\mathcal{H}_t)$, where $\mathcal{H}_t = \sigma(\{(t_i, k_i) : t_i < t\})$ is the information filtration containing the event history up to time t. For notational simplicity, we may later write $\lambda_k(t|\mathcal{H}_t)$ as $\lambda_k(t)$. The intensity $\lambda_k(t)$ represents the expected rate of events of type k occurring in an infinitesimal time interval $[t, t + dt)$, conditioned on the events that have already occurred by time t. Therefore, the probability that an event of type k occurs over $[t, t + dt)$ is approximately $\lambda_k(t)dt$, provided that dt is sufficiently small.

Whether for classical statistical TPPs or neural TPP models, the training objective is to learn the model parameters by minimizing the loss function, *Negative log-likelihood (NLL)*. Given the model's intensity function $\lambda_k(t)$ and an event sequence S, the corresponding NLL is defined as:

$$-\sum_{i=1}^{I} \log \lambda_{k_i}(t_i) + \sum_{k=1}^{K} \int_{t=0}^{T} \lambda_k(t)\, dt.$$

2.2. Classical Statistical TPPs

Several classical statistical models have been developed to characterize TPPs, where the intensity function usually follows a fixed parametric form.

Poisson Processes (Daley & Vere-Jones, 2007). The simplest model for TPPs is the Poisson Process, where the intensity function is defined as $\lambda_k(t)$, a function over time domain, indicating that the probability of an event occurring in a given time interval is independent of the history.

Hawkes Processes (Hawkes, 1971). The Hawkes Process captures the self-exciting nature of events in TPPs. The occurrence of an event not only depends on the baseline intensity but is also influenced by past events, making it suitable for modeling clustered or dependent events. The occurrence of an event causes a temporary increase in the

intensity, which decays over time:

$$\lambda_k(t) = \mu_k(t) + \sum_{i:t_i<t} \alpha_{k,k_i} g_{k,k_i}(t - t_i), \quad (1)$$

where $\mu_k(t) > 0$ is the baseline intensity, t_i denotes the times of past events, and $g(\cdot) > 0$ is a pre-specified decaying function that models the influence of past events on the current intensity. The exponential function $g_{k,k_i}(t - t_i) = \beta_{k,k_i} e^{-\beta_{k,k_i}(t-t_i)}$ is frequently used in this context. A major limitation of this formulation lies in its assumption that past events cannot inhibit occurrence of future events, which is unrealistic in complex real-life scenarios.

Self-Correcting Processes (Isham & Westcott, 1979). In self-correcting processes, the intensity function is typically modified by a negative feedback term that reduces the likelihood of subsequent events, thus correcting the system's behavior over time:

$$\lambda_k(t) = \Psi(\mu_k t - \sum_{i:t_i<t} \alpha_k),$$

where Ψ is a non-linear function, μ_k is the baseline coefficient, and α_k is the self-correction term that reduces the event intensity based on the accumulated history of events.

2.3. Neural TPPs

Neural TPP models extend the classical statistical models by parameterizing the intensity function through neural networks, rather than relying on a fixed parametric form. These models autoregressively generate future events conditioned on historical data. The most widely used neural TPPs can be broadly categorized as follows.

The first class is **RNN-based** (Du et al., 2016) or **Attention-based** (Zuo et al., 2020) TPPs. In this approach, the model computes the embedding of the i-th event (t_i, k_i) as $\mathbf{e}_i \in \mathbb{R}^D$ via an embedding layer. The hidden state $\mathbf{h}_i$ is updated based on the current event embedding $\mathbf{e}_i$ and the previous hidden state $\mathbf{h}_{i-1}$. The next event is then generated conditioned on $\mathbf{h}_i$:

$$t_{i+1}, k_{i+1} \sim \mathbb{P}_{\psi}(t_{i+1}, k_{i+1} \mid \mathbf{h}_i), \quad \mathbf{h}_i = f(\mathbf{h}_{i-1}, \mathbf{e}_i),$$

where f represents the encoder function, which can take various forms, such as recurrent architectures (e.g., LSTM, GRU) or more expressive transformer layers.

Another class, referred to as **ODE-based** (Jia & Benson, 2019), models the evolution of the hidden state in continuous time. Between event occurrences, the hidden state is updated continuously, typically governed by an ordinary differential equation (ODE). Specifically, the evolution of the hidden state from the $(i-1)$-th to the i-th event is given by:

$$\mathbf{h}_i = f(\mathbf{h}_{i-}, \mathbf{e}_i), \text{ where } \mathbf{h}_{i-} = f_{\text{ODE}}(\mathbf{h}_{i-1}, t_{i-1}, t_i).$$

A more complete list of existing neural models can be found in Appendix D.3.

3. Methodology

In this section, we propose the Residual TPP method to enhance the performance of existing models while also reducing the computational complexity. This method combines a simple statistical TPP with a neural TPP trained on the residuals, forming a simple yet powerful hybrid approach.

3.1. Motivation

Our motivation can be summarized as follows:

(i) Event stream data is complex due to irregular timing, heterogeneity, and the presence of noise or outliers, which may result from collection errors, unusual behavior, or rare events. Identifying and handling these outliers is crucial for accurate analysis but remains challenging. Classical TPPs typically rely on strong assumptions, such as the Poisson process or self-excitation, and use fixed parametric forms based on these assumptions. While these models are structurally simple and computationally efficient, they often perform poorly when applied to complex real-world data. As shown in Figure 1, the Hawkes process fits the data well most of the time, but it struggles to handle commission (added) and omission (removal) outliers (Liu & Hauskrecht, 2021b). Thus, we aim to propose a new approach that not only captures the self-excitation nature of data through the Hawkes process but also effectively handles outliers.

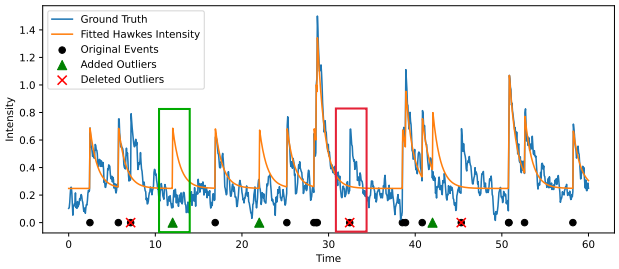

Figure 1. A simulation illustrating the impact of outliers on the Hawkes process. The ground truth intensity is a Hawkes process with added noise, where black dots represent real events generated from the ground truth. Green triangles and red crosses indicate randomly added and removed outliers, respectively. The fitted Hawkes process is estimated from the modified event sequence.

(ii) Recent advancements in neural TPPs have significantly improved performance over traditional methods. However, challenges remain: RNN-based TPPs inherit the intrinsic weaknesses of RNNs, i.e. their difficulty in capturing long-term dependencies. In contrast, more expressive attention-based TPPs, while offering enhanced modeling capabilities, suffer from high computational complexity and difficulties in training over long sequences. Therefore, our approach is designed to address the issue of lightweight modeling.

(iii) Our approach is inspired by the Seasonal-Trend Decomposition (STD) techniques in time series analysis, which decompose a series into trend, seasonal, and residual components, allowing models to better utilize periodic information and enhance prediction accuracy. Many popular models such as Autoformer (Wu et al., 2021), FEDformer (Zhou et al., 2022), and DLinear (Zeng et al., 2023), adopt the STD approach to decompose the time series and separately model seasonal and trend components. There is a significant body of research on STD methods. Classical approaches typically employ moving average kernels to estimate the trend component. More recently, several neural network-based methods have been proposed to model the periodicity in time series data (Fan et al., 2022; Lin et al., 2024a;b). However, despite similarities between event stream and time series data, current research on TPPs focuses mainly on modifying neural network architectures for improved performance. To the best of our knowledge, no studies have yet proposed decomposition methods for TPPs analogous to those used for time series. This gap arises from the unique challenges of event stream data, including **irregular timing**, **heterogeneous event types**, and **complex temporal dependencies**. Unlike time series, which are recorded at regular intervals and the observations are mostly continuous, event streams consist of asynchronous events that vary in type and frequency and event types take discrete values, often exhibiting high dimensionality and sparsity. Consequently, **defining the residual of a temporal point process is inherently difficult**. To address this gap, we introduce an innovative modeling and decomposition approach for temporal point processes.

3.2. Residual TPP

The Residual TPP method consists of three steps: First, a classical TPP is used to fit the statistical characteristics of the original sequence, such as periodicity and self-excitation, to obtain an intensity function. Next, the proposed Residual Events Decomposition (RED) technique is applied to separate the residual components of the sequence. Finally, a neural TPP is used to capture the intensity that remains unexplained by the classical model in the residual sequence. By combining the two intensity functions, the final TPP model is obtained. The overall algorithmic flow is illustrated in Figure 2 and Algorithm 1.

Periodic / Self-exciting patterns modeling. (Step 1) Event stream data often exhibit two key statistical properties: periodicity and self-excitation. Periodicity refers to the tendency of events to recur at regular intervals, commonly observed in fields like neuroscience and economics, where events follow repetitive cycles. Self-excitation refers to the phenomenon where the occurrence of an event increases the likelihood of future events. This is important in modeling phenomena such as earthquakes, financial market crashes, or social media activity, where an event can trigger subsequent activity. In the view of statistical terminology, periodicity and self-excitation correspond to the first-order and second-order statistics of event stream data. These are the main features that should be captured in TPP modeling. Further explanations can be found in Appendix B.

Standard statistical TPP model, e.g. the Hawkes process, can be well-suited to model both periodicity and self-excitation, with its intensity function taking the form described in (1). To capture periodicity, the baseline function $\mu_k(t)$ can be selected to exhibit periodic behavior. To model self-excitation, $g(\cdot)$ is typically chosen to be a non-negative function that decays over time, such as an exponential decay or a power-law decay, reflecting the fact that recent events have a stronger impact on future events but this influence diminishes with time. Therefore, in order to capture the key statistical properties of the data in a simple and efficient manner, we first fit the Hawkes process intensity $\lambda_k^{(1)}(t)$ to the event sequences, where the overall intensity function is given by $\lambda^{(1)}(t) = \sum_{k=1}^{K} \lambda_k^{(1)}(t)$.

Residual Events Decomposition (RED). (Step 2) Next, we provide a solution to define the residual of the temporal point process data. Our objective is to evaluate the performance of the intensity function $\lambda_k^{(1)}(t)$ obtained from the statistical TPP model, and to identify the data points where the intensity fit is poor. Therefore, we aim to construct a weight function such that events well captured by $\lambda^{(1)}(t)$ receive larger weights, allowing us to filter out the residual events based on these weights. To achieve this, a specific weight function is given as follows.

We denote $\boldsymbol{\theta} := \{\theta_k\}_{k=1}^{K}$ as the complete parameter set in $\lambda_k^{(1)}(t) = \text{Hawkes}(\mathbf{S}; \theta_k)$, where θ_k is the parameter set for event type k. For example, in the case of Hawkes process, $\theta_k := [\mu_k(\cdot), g_{k,k'}(\cdot)], \quad k' = 1, \cdots, K$. Given a set of parameters $\boldsymbol{\theta}$ obtained in Step 1, we define the weight of the i-th event in sequence S as follows:

$$W_i(S; \boldsymbol{\theta}) = \phi'_{\rho_1,\rho_2}\left(\int_{t_{i-1}}^{t_i} \sum_{k=1}^{K} \lambda_k^{(1)}(u)du - 1\right), \quad (2)$$

where $t_0 = 0$, $i = 1, \cdots, I$, and $\phi'_{\rho_1,\rho_2}(x)$ is defined as

$$\phi'_{\rho_1,\rho_2}(x) = \begin{cases} \phi'(x/\rho_2), & x \geq 0, \\ \phi'(x'/\rho_1), & -1 \leq x < 0. \end{cases} \quad (3)$$

Here, $\phi'(x)$ denotes the derivative function of an influence function $\phi(x)$, which, for $x \geq 0$, takes the following form:

$$\phi'(x) = \begin{cases} \frac{1+x}{1+x+x^2/2} & 0 \leq x \leq a, \\ \phi'(a) \cdot \frac{(b-x)^2}{(b-a)^2} & a < x \leq b, \\ 0 & x > b. \end{cases} \quad (4)$$

Additionally, $\phi'(x) = \phi'(x')$ for $-1 \leq x < 0$, where $x' :=$

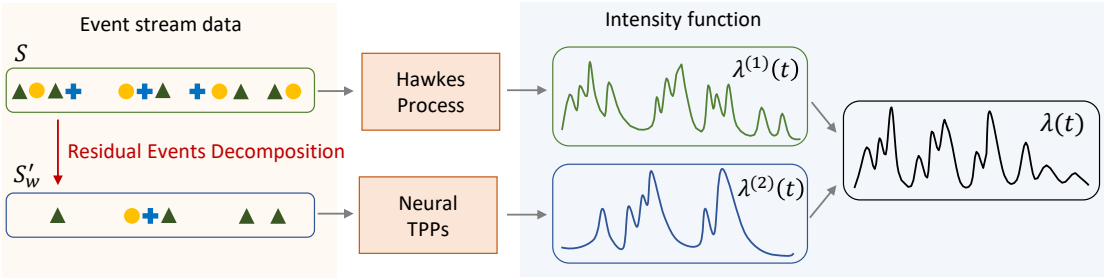

Figure 2. Algorithm flowchart of the Residual TPP. The event stream data here consists of three event types, represented by triangles, circles, and plus signs, respectively.

$\{x' | (x'+1)\exp(-x'-1) = (x+1)\exp(-x-1), x' \geq 0\}$. ρ_1, ρ_2, a and b are positive tuning parameters.

It is evident that $\phi'(x)$ attains its maximum at $x = 0$ and has compact support. The influence function $\phi(x)$ is increasing and twice continuously differentiable. Due to these properties, $W_i(S;\boldsymbol{\theta})$ approaches 1 when $\int_{t_{i-1}}^{t_i} \lambda^{(1)}(u)\,du$ is close to 1, while $W_i(S;\boldsymbol{\theta})$ will approach 0 when the integral deviates significantly from 1. The expression $\int_{t_{i-1}}^{t_i} \lambda^{(1)}(u)\,du$ represents the average number of events predicted by the Hawkes process within the time interval $(t_{i-1}, t_i]$.

Furthermore, based on the time transformation theory, if the event sequence S is generated by a point process with intensity function $\lambda^*(t)$, then $\int_{t_{i-1}}^{t_i} \lambda^*(u)\,du \sim \text{Exp}(1)$. In other words, weight $W_i(S;\boldsymbol{\theta})$ is close to 1 when the number of events predicted by the Hawkes process closely matches the actual observed number of events, indicating that the model accurately captures event occurrences within this interval. This also suggests that the event occurrence within this time interval is relatively predictable, as the model can accurately forecast the probability of an event occurring.

After obtaining the weight for each event, the residual component of sequence S can be identified by applying a threshold w $(0 \leq w \leq 1)$ as follows:

$$S'_w = \{(t_i, k_i) : W_i(S;\boldsymbol{\theta}) \leq w\}.$$

Residual modeling. (Step 3) After identifying the residual events, we train a neural TPP model on the residual sequence dataset $\mathbf{S}'_w$, yielding the following intensity function:

$$\lambda_k^{(2)}(t) = \text{Neural TPP}(\mathbf{S}'_w; \psi_k), \quad k = 1, ..., K,$$

where the Neural TPP represents an arbitrarily existing model (parametrized by ψ_k's), such as RNN-based or attention-based models, among others.

Combined Intensity. Due to the superposition property of temporal point processes, the final intensity function for event type k is given by the combination of the Hawkes process and the neural TPP for residual events:

$$\lambda_k(t) = (1-\alpha)\lambda_k^{(1)}(t) + \lambda_k^{(2)}(t),$$

and the overall intensity function is $\lambda(t) = \sum_{k=1}^{K} \lambda_k(t)$. Here, $\alpha = \frac{|\mathbf{S}'_w|}{|\mathbf{S}|}$ represents the proportion of residual events in $\mathbf{S}'_w$ relative to the total number of events in dataset $\mathbf{S}$.

The intensity function in the proposed model is a higher-level concept, encompassing both classical statistical TPPs and neural TPPs. Specifically, when the threshold $w = 0$, $\mathbf{S}'_w = \emptyset$, indicating that we believe the statistical model has adequately fit the original events, and no residual data is selected for training the neural TPP model. In this case, $\alpha = 0$ and $\lambda_k(t) = \lambda_k^{(1)}(t)$, making the model equivalent to a Hawkes process. On the other hand, when the threshold $w = 1$, we have $\mathbf{S}'_w = \mathbf{S}$, meaning that the neural TPP model is trained on the entire original sequences. Here, $\alpha = 1$ and $\lambda_k(t) = \lambda_k^{(2)}(t)$, which corresponds to a neural TPP. When $0 < w < 1$, the resulting intensity function is a combination of the Hawkes process and the neural TPP on the residual data, capturing both the self-excitation in the original sequence and the unexplained intensity from the residual data, thereby yielding a more powerful model.

In practical applications, α can also be treated as a hyperparameter to be tuned, allowing for a more flexible balance between the statistical properties modeled by the classical TPP and the residuals captured by the neural TPP. This balance can lead to the improved model performance.

Prediction. Given a prefix of the event sequence $S_{[0,t_{i-1}]} = \{(t_1, k_1), \ldots, (t_{i-1}, k_{i-1})\}$, we want to predict the time and type of the next event. The probability density of the next event time t_i is given by

$$p_i(t) = P(t_i = t \mid \mathcal{H}_{t_{i-1}}) = \lambda(t)\exp\left(-\int_{t_{i-1}}^{t} \lambda(s)\,ds\right).$$

To minimize the expected L_2 loss in predicting the event time, we select the expected value:

$$\hat{t}_i = E[t_i \mid \mathcal{H}_{t_{i-1}}] = \int_{t_{i-1}}^{\infty} t p_i(t)\,dt. \tag{5}$$

Algorithm 1 The Overall Pseudocode of Residual TPP

INPUT: Event dataset $\mathbf{S} = \{S_n\}_{n=1}^N$, influence function ϕ
OUTPUT: Intensity function $\lambda_k(t)$

Initialize parameters $\boldsymbol{\theta} = \{\theta_k\}_{k=1}^K$, $\boldsymbol{\psi} = \{\psi_k\}_{k=1}^K$
—- Step 1 —-
Fit the Hawkes process with full data $\mathbf{S}$ and obtain $\hat{\boldsymbol{\theta}}$
$\lambda_k^{(1)}(t) \leftarrow \text{Hawkes}(\mathbf{S}; \hat{\theta}_k)$
—- Step 2 —-
for S_n in $\mathbf{S}$ **do**
  Initialize the residual set $S'_{n,w} = \emptyset$
  for $(t_{n,i}, k_{n,i})$ in sequences S_n **do**
    $W_i(S_n; \hat{\boldsymbol{\theta}}) \leftarrow \phi'_{\rho_1,\rho_2}\left(\int_{t_{n,i-1}}^{t_{n,i}} \sum_{k=1}^K \lambda_k^{(1)}(u)du - 1\right)$
    if $W_i(S_n; \hat{\boldsymbol{\theta}}) < w$ **then**
      $S'_{n,w} \leftarrow S'_{n,w} \cup \{(t_{n,i}, k_{n,i})\}$
    end if
  end for
  Obtain the residual event dataset $\mathbf{S}'_w = \{S'_{n,w}\}_{n=1}^N$
end for
—- Step 3 —-
Compute $\alpha \leftarrow \frac{|\mathbf{S}'_w|}{|\mathbf{S}|}$
Fit a neural TPP model to the residual $\mathbf{S}'_w$ to obtain $\hat{\boldsymbol{\psi}}$
$\lambda_k^{(2)}(t) \leftarrow \text{Neural TPP}(\mathbf{S}'_w; \hat{\psi}_k)$
$\lambda_k(t) \leftarrow (1-\alpha)\lambda_k^{(1)}(t) + \lambda_k^{(2)}(t)$

Given the actual next event time t_i, the most likely type of the next event is given by

$$\hat{k}_i = \text{argmax}_{k\in[K]} \frac{\lambda_k(t_i)}{\lambda(t_i)}. \tag{6}$$

3.3. Theoretical Justification of the Weight Function

As recently shown in Zhang et al., 2025, the influence function defined in (4) enjoys the "unbiasedness" properties.

Proposition 3.1. *When X follows the standard exponential distribution Exp(1), it holds that* $\mathbb{E}\left[(X-1)\cdot\phi'(X-1)\right] = 0$.

By this property, we can show that the expected gradient evaluated at the true model parameters remains asymptotically unbiased after incorporating the weight function when there is no event contamination and $T \to \infty$. To be more mathematically formal, for any TPP model with intensity function $\lambda_{\boldsymbol{\theta}}(t)$ parametrized by $\boldsymbol{\theta}$, the weighted gradient is defined as

$$\varrho(\boldsymbol{\theta}) = \frac{1}{T}\sum_{i=1}^I W_i(S;\boldsymbol{\theta})\nabla_{\boldsymbol{\theta}}\Big(\log\lambda_{\boldsymbol{\theta}}(t_{i-1}) - \int_{t_{i-1}}^{t_i}\lambda_{\boldsymbol{\theta}}(t)dt\Big),$$

where $W_i(S;\boldsymbol{\theta}) = \phi'_{\rho_1,\rho_2}(\int_{t_{i-1}}^{t_i}\lambda_{\boldsymbol{\theta}}(u)du - 1)$.

Theorem 3.2. *When $\rho_1 = \rho_2$, $\nabla_{\boldsymbol{\theta}}\log\lambda_{\boldsymbol{\theta}}(t)$ and $\lambda_{\boldsymbol{\theta}}(t)$ are bounded at the true $\boldsymbol{\theta}^*$, it holds that*

$$\left|\mathbb{E}[\varrho(\boldsymbol{\theta})]\big|_{\boldsymbol{\theta}=\boldsymbol{\theta}^*}\right| = O(\frac{1}{T}). \tag{7}$$

Note that $\mathbb{E}[\varrho(\boldsymbol{\theta})]$ is a vector and (7) holds elementwisely. The theorem says that, although $\int_{t_{i-1}}^{t_i}\lambda_k(u)du$ is left-skewed, the careful design of $\phi(x)$ ensures that the proposed weight function carefully balances the left and right tails of the integral of the intensity function, thereby allowing them to have equal impacts on the loss function. Theorem 3.2 is the generalization of Theorem 3 in Zhang et al., 2025, where they only allow $\lambda_{\boldsymbol{\theta}}(t)$ to take some specific forms. Here $\lambda_{\boldsymbol{\theta}}(t)$ can take any parametric form, including neural network-based models as well.

Therefore, Step 2 of Algorithm 1 can safely remove those events which are well captured by the Hawkes processes. The remaining events in $\mathbf{S}'_w$ are viewed as unexpected outliers that may not have statistical interpretations (i.e., periodic baseline events or self-exciting events).

4. Experiments

In this section, we present the experimental results of our method on six mainstream real-world TPP benchmarks and three synthetic datasets with six baseline models. Our code is publicly available at https://github.com/ruoxinyuan/ResidualTPP.

4.1. Experimental Setup

Datasets. We evaluate our method on six real-world benchmark datasets: *MIMIC-II* (Johnson et al., 2016), *Retweet* (Zhou et al., 2013), *Earthquake* (Xue et al., 2024), *StackOverflow* (Leskovec & Krevl, 2014), *Amazon* (Ni, 2018) and *Volcano* (Xue et al., 2024). The MIMIC-II dataset is available at the public GitHub repository[1], and the others can be accessed via the EasyTPP library[2]. Detailed descriptions of these datasets are provided in Appendix D.2.

In addition to real-world benchmarks, we construct three synthetic datasets for controlled evaluation. (i) *Poisson-based*: We generate a non-homogenous Possion process with five event types, each with a different periodic triangular function for $\lambda_k^{(1)}$, and set residual events to follow $\lambda_k^{(2)} = 0.1$, a homogeneous Poisson process. The superposition of these processes yields the Poisson-based dataset. (ii) *AttNHP-based*: We use the AttNHP (Yang et al., 2022) to model $\lambda^{(1)}$, with the residuals again modeled by a homogeneous Poisson process $\lambda_k^{(2)} = 0.1$. (iii) *Possion + AttNHP*:

[1] https://github.com/hongyuanmei/neurawkes
[2] https://github.com/ant-research/EasyTemporalPointProcess

Table 1. Performance of all the methods on the goodness-of-fit task. The first row presents the log-likelihood on the test set for the multivariate Hawkes process. For each baseline model, the first row indicates the performance of the baseline neural TPP, while the second row shows the performance of the corresponding Residual TPP. Higher scores indicate better performance.

MODEL	GOODNESS-OF-FIT (LOG-LIKELIHOOD)					
	MIMIC-II	RETWEET	EARTHQUAKE	STACKOVERFLOW	AMAZON	VOLCANO
MHP	-2.839	-13.71	-4.155	-2.866	-0.534	0.983
RMTPP	-2.626	-4.382	-4.643	-2.844	-2.389	-3.776
RES RMTPP	**-2.045**	**-3.240**	**-3.689**	**-0.864**	**-1.419**	**-3.548**
NHP	-2.031	-4.146	-2.389	-2.613	-2.199	0.261
RES NHP	**-1.825**	**-3.881**	**-1.930**	**-0.712**	**-1.303**	**0.435**
SAHP	-4.672	-4.564	-3.338	-3.867	-2.370	0.128
RES SAHP	**-4.488**	**-4.493**	**-3.335**	**-2.374**	**-1.548**	**0.256**
THP	-2.048	**-4.548**	-3.498	-2.747	-2.365	-0.068
RES THP	**-2.040**	-4.597	**-3.415**	**-2.467**	**-1.611**	**0.233**
ATTNHP	-2.500	-4.729	**-2.896**	-2.685	-2.376	-0.303
RES ATTNHP	**-2.197**	**-4.576**	-3.147	**-2.482**	**-1.383**	**-0.050**
ODETPP	-1.855	-4.527	**-2.203**	-2.492	-2.468	-0.078
RES ODETPP	**-1.371**	**-4.491**	-2.340	**-2.467**	**-1.394**	**0.321**

We use the same periodic non-homogenous Possion process for $\lambda^{(1)}$ and AttNHP for $\lambda^{(2)}$. Summary statistics for these synthetic datasets are provided in Table 8.

Each dataset is split into training, validation, and test sets. The validation set is for hyperparameter tuning, and the test set is for model evaluation. Due to the wide distribution range of event times and the sparsity of events in the Volcano dataset, we additionally scale the time series by dividing it by one thousand.

Baselines. In the experiments, we integrate the multivariate Hawkes process (MHP) with several neural TPP models to derive the corresponding Residual TPP. We employ all intensity-based neural TPPs from the EasyTPP library, for integration into the Residual TPP and as baselines for comparison. The models include two RNN-based models: the Recurrent Marked Temporal Point Process (**RMTPP**, Du et al. (2016)) and the Neural Hawkes Process (**NHP**, Mei & Eisner (2017)); three attention-based models: the Self-Attentive Hawkes Process (**SAHP**, Zhang et al. (2020a)), the Transformer Hawkes Process (**THP**, Zuo et al. (2020)), and the Attentive Neural Hawkes Process (**AttNHP**, Yang et al. (2022)); and one TPP with a hidden state governed by a neural jump SDE: the simplified version of Neural Spatio-Temporal Point Process (**ODETPP**, Chen et al. (2021)). More details on the baseline models are provided in Appendix D.3. As clarified, the original FullyNN model (Omi et al., 2019), with a fully neural network-based intensity, does not support multi-type event sequences and exhibit worse fitness than other neural competitors across all datasets. Consequently, we omit it from the main text, with specific results provided in Appendix E.2.

Implementation details. For simplicity, we model the event stream data using a Hawkes process with a fixed baseline intensity and an exponential decay function to capture the self-excitation property:

$$\lambda_k^{(1)}(t) = \mu_k + \sum_{i:t_i<t} \alpha_{k,k_i}\beta_{k,k_i}e^{-\beta_{k,k_i}(t-t_i)}.$$

In our experiments, we implement this using Tick[3] (Bacry et al., 2017). A more comprehensive treatment of periodicity will be explored in our future work.

We follow the architecture of all neural TPP models as outlined in the original implementations of the respective papers, utilizing the code from EasyTPP (Xue et al., 2024). To ensure a fair comparison, the training parameters and procedures for the corresponding neural TPP models trained on the original data are kept consistent with those for models trained on the residual events filtered by RED.

Evaluation. After obtaining the Residual TPPs and the corresponding baseline neural TPPs, we evaluate the models using three standard metrics on the test set:

- Goodness-of-fit: We evaluate the log-probability assigned to the test set by the models, with higher values indicating better model performance in terms of fitting the data.
- Next event-time prediction: We predict the next event's time using the minimum Bayes risk principle, based solely on preceding events, as described in (5). The accuracy is assessed using the Root Mean Squared Error (RMSE).

[3] https://github.com/X-DataInitiative/tick

Table 2. Performance of all methods on next-event time prediction and next-event type prediction across six datasets. Lower scores indicate better performance. As noted, the Volcano dataset is not applicable for the type prediction task, as it contains only one event type.

Model	Prediction Performance (Time RMSE / Type Error Rate)					
	MIMIC-II	Retweet	Earthquake	StackOverflow	Amazon	Volcano
MHP	0.925/27.9%	21.98/53.5%	1.475/60.6%	1.619/57.5%	1.042/70.4%	4.223/N.A.
RMTPP	0.998/37.8%	20.68/44.6%	1.956/52.9%	1.334/57.5%	0.613/68.4%	4.696/N.A.
Res RMTPP	**0.915/26.7%**	**19.31/44.5%**	**1.420/52.7%**	**1.315/57.5%**	**0.504/67.5%**	**3.709**/N.A.
NHP	1.010/26.7%	21.97/42.3%	1.910/53.9%	1.411/56.2%	0.611/67.2%	3.906/N.A.
Res NHP	**0.913/20.9%**	**19.14/40.9%**	**1.416/52.8%**	**1.383/56.1%**	**0.514/66.1%**	**3.854**/N.A.
SAHP	0.971/23.8%	22.13/44.8%	1.463/54.1%	1.348/55.6%	0.605/68.3%	3.998/N.A.
Res SAHP	**0.935/18.0%**	**19.85/42.0%**	**1.455/53.3%**	**1.245/55.4%**	**0.498/67.0%**	**3.940**/N.A.
THP	1.129/35.5%	22.01/45.4%	1.857/54.7%	1.401/56.1%	0.622/66.8%	3.945/N.A.
Res THP	**0.930/27.9%**	**19.33/41.5%**	**1.403/52.8%**	**1.376/55.7%**	**0.515/65.7%**	**3.910**/N.A.
AttNHP	1.030/**35.4%**	21.78/43.1%	1.822/54.5%	1.384/57.1%	0.640/66.4%	4.032/N.A.
Res AttNHP	**0.932**/36.0%	**19.41/42.9%**	**1.413/52.8%**	**1.369/57.0%**	**0.633/65.4%**	**3.918**/N.A.
ODETPP	1.416/22.1%	22.48/43.2%	2.396/56.0%	1.469/56.0%	0.697/67.6%	3.927/N.A.
Res ODETPP	**0.934/19.2%**	**19.45/42.7%**	**1.412/53.0%**	**1.378/55.6%**	**0.643/67.3%**	**3.865**/N.A.

Table 3. End-to-end training time. "MHP + RED" reports the average runtime of Steps 1 and 2 over 10 independent trials. For each Res TPP, the result reflects the total time cost of the 3-step procedure (MHP + RED + neural TPP). All training was conducted on a CPU.

Model	End-to-end Runtime (seconds)					
	MIMIC-II	Retweet	Earthquake	StackOverflow	Amazon	Volcano
MHP+RED	1.42	0.27	0.11	0.84	1.49	0.02
RMTPP	**9.70**	195.3	26.00	71.20	95.75	34.20
Res RMTPP	9.72	**188.7**	**20.01**	**70.44**	**95.19**	**26.42**
NHP	34.70	492.6	25.32	206.3	256.6	77.05
Res NHP	**29.52**	**441.9**	**20.57**	**190.8**	**245.1**	**54.47**
SAHP	128.5	498.4	24.90	498.0	505.8	37.30
Res SAHP	**113.4**	**434.9**	**16.96**	**461.9**	**450.3**	**24.42**
THP	**10.45**	1183	19.20	81.20	257.7	39.90
Res THP	10.67	**1029**	**17.77**	**76.44**	**244.8**	**29.82**
AttNHP	68.60	9475	162.2	1924	6375	1093
Res AttNHP	**52.92**	**7195**	**143.1**	**1863**	**5646**	**657.5**
ODETPP	9.96	106.4	42.00	243.6	196.4	51.96
Res ODETPP	**9.64**	**102.7**	**37.43**	**224.2**	**192.1**	**40.88**

- Next event-type prediction: We forecast the corresponding event type using both its actual time and the preceding events according to (6). The prediction accuracy is measured by the error rate.

4.2. Main Results

Table 1 reports the log-likelihood of the models on each test set. It reveals that Residual TPP fits the data well and significantly outperforms the corresponding neural TPPs across all datasets. This highlights our model's capability to capture complex real-world intensity dynamics from diverse domains. The results for next event-time and event-type prediction, presented in Table 2, further confirm the superior performance of our method. Residual TPP achieves a notable improvement in predictive accuracy. Table 3 evaluates the computational efficiency by reporting the training time for the baseline neural models and their corresponding Residual TPPs. Across all datasets, Residual TPP consistently exhibits lower training times compared to the baseline models. These experimental results demonstrate that the proposed Residual TPP is a powerful yet lightweight approach for modeling event streams. It can be integrated with various types of neural TPPs, enhancing both performance and computational efficiency.

4.3. Simulation Analysis

To further validate the robustness of RED, we generate synthetic datasets under various configurations of primary

Table 4. Performance comparison on synthetic datasets. *Goodness-of-fit* is measured by log-likelihood (L-L; higher is better). *Prediction* performance is evaluated using time RMSE and type error rate (lower is better).

MODEL	POSSION-BASED		ATTNHP-BASED		POSSION+ATTNHP	
	L-L	TIME/TYPE	L-L	TIME/TYPE	L-L	TIME/TYPE
MHP	-0.132	0.298/62.2%	-1.435	0.329/79.4%	-0.203	0.257/68.1%
RMTPP	-1.030	0.337/63.2%	-1.277	0.373/78.5%	-0.788	0.271/68.6%
RES RMTPP	**-0.531**	**0.305/62.3%**	**-0.664**	**0.334/77.9%**	**-0.027**	**0.237/67.8%**
NHP	-1.023	0.343/62.1%	-1.267	0.374/77.4%	-0.776	0.277/67.8%
RES NHP	**-0.527**	**0.301/62.0%**	**-0.661**	**0.334/77.1%**	**-0.009**	**0.237**/67.8%
SAHP	-1.033	0.330/62.1%	-1.267	0.376/77.5%	-0.789	0.282/67.8%
RES SAHP	**-0.556**	**0.303**/62.1%	**-0.660**	**0.335/77.3%**	**-0.002**	**0.238**/67.8%
THP	-1.026	0.336/62.6%	-1.270	0.374/77.6%	-0.785	0.266/67.8%
RES THP	**-0.566**	**0.300/62.0%**	**-0.655**	**0.331**/77.6%	**0.004**	**0.234**/67.8%
ATTNHP	-1.026	0.337/62.1%	-1.266	0.374/77.3%	-0.789	0.269/67.8%
RES ATTNHP	**-0.534**	**0.302**/62.1%	**-0.658**	**0.336/77.2%**	**0.001**	**0.237**/67.8%
ODETPP	-1.028	0.339/62.1%	-1.267	0.382/77.6%	-0.789	0.270/67.8%
RES ODETPP	**-0.565**	**0.302**/62.1%	**-0.663**	**0.332**/77.6%	**-0.001**	**0.238**/67.8%

intensities $\lambda^{(1)}$ and residual intensities $\lambda^{(2)}$, and evaluate the effectiveness of the RED technique across three distinct settings, as illustrated in Section 4.1.

We compare the performance of Residual TPP against baseline neural TPPs on these synthetic datasets. As shown in Table 4, Residual TPP consistently improves the performance of neural TPPs through RED, even when the underlying true signal does not follow a Hawkes process. This demonstrates that the residuals identified by RED are agnostic to the true data-generating process. RED quantifies how well a given statistical model (e.g., Hawkes process) explains the observed events through its weight function. Even if the true process deviates significantly from the assumed model, RED can isolate the unexplained residuals for refinement by a neural TPP. This is conceptually similar to residual learning in deep neural networks, where residuals represent deviations from a simpler base function, regardless of its exact form.

5. Conclusion

In this study, we introduce the Residual TPP, a novel framework that bridges the gap between simple statistical TPPs and expressive neural TPPs through Residual Events Decomposition (RED), providing a unified solution that combines the strengths of both. Extensive experiments demonstrate that Residual TPP significantly outperforms existing models, offering a robust and scalable approach for event stream data analysis. We also validate the effectiveness of RED as a novel decomposition method compatible with various existing TPP architectures.

This framework establishes an extensible foundation for future research. First, the baseline intensity and decay function in the Hawkes process can be adjusted to better capture specific data characteristics like periodicity. Additionally, decomposition methods for TPPs remain an underexplored area. As the first decomposition technique, RED provides valuable insights for developing further methods, similar to STD in times series analysis, to simplify computational procedure and enhance model performance.

Overall, the flexibility of RED as a plug-and-play module makes Residual TPP adaptable to various TPP models and tasks. Residual TPP offers a lightweight but powerful solution for event stream analysis. This work not only advances event stream analysis capabilities but also opens new directions for developing hybrid TPP frameworks through innovative decomposition strategies. Moreover, beyond event streams, the RED technique could be similarly extended to other complex, discrete, and irregular data types.

Acknowledgements

The authors thank the anonymous reviewers for their constructive comments on improving the quality of our paper. Guanhua Fang is partly supported by the National Natural Science Foundation of China (nos. 12301376) and Shanghai Educational Development Foundation (23CGA02).

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning related to temporal point processes and continuous time event streaming analyses. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted here.

References

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

Bacry, E., Bompaire, M., Gaïffas, S., and Poulsen, S. Tick: a Python library for statistical learning, with a particular emphasis on time-dependent modeling. *ArXiv e-prints*, July 2017.

Bae, W., Ahmed, M. O., Tung, F., and Oliveira, G. L. Meta temporal point processes. In *The International Conference on Learning Representations (ICLR)*, 2023.

Biloš, M., Charpentier, B., and Günnemann, S. Uncertainty on asynchronous time event prediction. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Chen, R. T. Q., Rubanova, Y., Bettencourt, J., and Duvenaud, D. K. Neural ordinary differential equations. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

Chen, R. T. Q., Amos, B., and Nickel, M. Neural Spatio-Temporal Point Processes. In *International Conference on Learning Representations*, 2021.

Daley, D. and Vere-Jones, D. An introduction to the theory of point processes. Vol. I: Elementary theory and methods. Vol. 1, 01 2003. doi: 10.1007/b97277.

Daley, D. J. and Vere-Jones, D. *An Introduction to the Theory of Point Processes, Volume II: General Theory and Structure*, volume 2. Springer, 2007.

Du, N., Dai, H., Trivedi, R., Upadhyay, U., Gomez-Rodriguez, M., and Song, L. 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*, KDD '16, pp. 1555–1564, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450342322.

Fan, W., Zheng, S., Yi, X., Cao, W., Fu, Y., Bian, J., and Liu, T.-Y. DEPTS: Deep expansion learning for periodic time series forecasting. In *International Conference on Learning Representations*, 2022.

Farajtabar, M., Wang, Y., Gomez-Rodriguez, M., Li, S., Zha, H., and Song, L. Coevolve: A joint point process model for information diffusion and network evolution. *Journal of Machine Learning Research*, 18(41):1–49, 2017.

Fox, E. W., Schoenberg, F. P., and Gordon, J. S. Spatially inhomogeneous background rate estimators and uncertainty quantification for nonparametric Hawkes point process models of earthquake occurrences. *The Annals of Applied Statistics*, 10(3):1725 – 1756, 2016. doi: 10.1214/16-AOAS957.

Hasbrouck, J. Measuring the information content of stock trades. *The Journal of Finance*, 46(1):179–207, 1991.

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

Isham, V. and Westcott, M. A self-correcting point process. *Stochastic Processes and their Applications*, 8:335–347, 1979.

Jia, J. and Benson, A. R. Neural jump stochastic differential equations. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Johnson, A. E., Pollard, T. J., Shen, L., Lehman, L.-w. H., Feng, M., Ghassemi, M., Moody, B., Szolovits, P., Celi, L. A., and Mark, R. G. MIMIC-III, a freely accessible critical care database. *Scientific Data*, 3(1):160035, May 2016.

Kingma, D. P. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kingman, J. F. C. *Poisson Processes: Oxford Studies In Probability. 3*. Oxford University Press, 12 1992.

Leskovec, J. and Krevl, A. Snap Datasets: Stanford large network dataset collection, 2014.

Lin, S., Lin, W., Hu, X., Wu, W., Mo, R., and Zhong, H. CycleNet: Enhancing time series forecasting through modeling periodic patterns. In *Thirty-eighth Conference on Neural Information Processing Systems*, 2024a.

Lin, S., Lin, W., Wu, W., Chen, H., and Yang, J. SparseTSF: Modeling long-term time series forecasting with 1k parameters. In *Forty-first International Conference on Machine Learning*, 2024b.

Liu, S. and Hauskrecht, M. Event outlier detection in continuous time. In Meila, M. and Zhang, T. (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 6793–6803. PMLR, 18–24 Jul 2021a.

Liu, S. and Hauskrecht, M. Event outlier detection in continuous time. In *International Conference on Machine Learning*, pp. 6793–6803. PMLR, 2021b.

Mei, H. and Eisner, J. M. The Neural Hawkes Process: A Neurally Self-Modulating Multivariate Point Process. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Ni, J. Amazon Product Review Dataset, 2018. URL https://nijianmo.github.io/amazon/.

Ogata, Y. Statistical models for earthquake occurrences and residual analysis for point processes. *Journal of the American Statistical Association*, 83(401):9–27, 1988.

Omi, T., ueda, n., and Aihara, K. Fully neural network based model for general temporal point processes. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. PyTorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Shchur, O., Biloš, M., and Günnemann, S. Intensity-free learning of temporal point processes. In *International Conference on Learning Representations*, 2020.

Shchur, O., Turkmen, A. C., Januschowski, T., Gasthaus, J., and Günnemann, S. Detecting anomalous event sequences with temporal point processes. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 13419–13431. Curran Associates, Inc., 2021.

Williams, A., Degleris, A., Wang, Y., and Linderman, S. Point process models for sequence detection in high-dimensional neural spike trains. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 14350–14361. Curran Associates, Inc., 2020.

Wu, H., Xu, J., Wang, J., and Long, M. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 22419–22430. Curran Associates, Inc., 2021.

Xu, H., Farajtabar, M., and Zha, H. Learning granger causality for hawkes processes. In Balcan, M. F. and Weinberger, K. Q. (eds.), *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 1717–1726, New York, New York, USA, 20–22 Jun 2016. PMLR.

Xue, S., Shi, X., Chu, Z., Wang, Y., Hao, H., Zhou, F., Jiang, C., Pan, C., Zhang, J. Y., Wen, Q., Zhou, J., and Mei, H. EasyTPP: Towards Open Benchmarking Temporal Point Processes. In *International Conference on Learning Representations (ICLR)*, 2024.

Yang, C., Mei, H., and Eisner, J. Transformer Embeddings of Irregularly Spaced Events and Their Participants. In *International Conference on Learning Representations*, 2022.

Zeng, A., Chen, M., Zhang, L., and Xu, Q. Are transformers effective for time series forecasting? *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(9):11121–11128, Jun. 2023.

Zhang, Q., Lipani, A., Kirnap, O., and Yilmaz, E. Self-Attentive Hawkes Process. In III, H. D. and Singh, A. (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 11183–11193. PMLR, 13–18 Jul 2020a.

Zhang, S., Zhou, C., Liu, Y., Zhang, P., Lin, X., and Ma, Z.-M. Neural jump-diffusion temporal point processes. In *International Conference on Machine Learning*, 2024.

Zhang, W., Panum, T., Jha, S., Chalasani, P., and Page, D. CAUSE: Learning Granger causality from event sequences using attribution methods. In III, H. D. and Singh, A. (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 11235–11245. PMLR, 13–18 Jul 2020b.

Zhang, Y., Cao, D., and Liu, Y. Counterfactual neural temporal point process for estimating causal influence of misinformation on social media. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A. (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 10643–10655. Curran Associates, Inc., 2022.

Zhang, Y., Fang, G., and Yu, W. Learning under commission and omission event outliers. *arXiv preprint arXiv:2501.13599*, 2025.

Zhou, K., Zha, H., and Song, L. Learning Triggering Kernels for Multi-dimensional Hawkes Processes. In Dasgupta, S. and McAllester, D. (eds.), *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pp. 1301–1309, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR.

Zhou, T., Ma, Z., Wen, Q., Wang, X., Sun, L., and Jin, R. FEDformer: Frequency enhanced decomposed transformer for long-term series forecasting. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., and Sabato, S. (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 27268–27286. PMLR, 17–23 Jul 2022.

Zuo, S., Jiang, H., Li, Z., Zhao, T., and Zha, H. Transformer Hawkes process. In III, H. D. and Singh, A. (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 11692–11702. PMLR, 13–18 Jul 2020.

A. Technical Proof

Proof of Theorem 3.2

Suppose $\boldsymbol{\theta}$ is a n-dimensional parameter vector, i.e., $\boldsymbol{\theta} = (\theta_1, ..., \theta_n)$. By recalling the definition of $\varrho(\boldsymbol{\theta})$, the gradient can be written as

$$\begin{aligned}
&\varrho(\boldsymbol{\theta}) \\
=&\begin{pmatrix}
\sum_{i=1}^{I} \phi'\left(\int_{t_{i-1}}^{t_i} \lambda_{\boldsymbol{\theta}}(u)du - 1\right) \cdot \left(\frac{\lambda'_{\boldsymbol{\theta},1}(t_{i-1})}{\lambda_{\boldsymbol{\theta}}(t_{i-1})} - \int_{t_{i-1}}^{t_i} \lambda'_{\boldsymbol{\theta},1}(x)dx\right) - \phi'\left(\int_0^{t_1} \lambda_{\boldsymbol{\theta}}(u)du - 1\right) \int_0^{t_1} \lambda'_{\boldsymbol{\theta},1}(x)dx \\
\vdots \\
\sum_{i=1}^{I} \phi'\left(\int_{t_{i-1}}^{t_i} \lambda_{\boldsymbol{\theta}}(u)du - 1\right) \cdot \left(\frac{\lambda'_{\boldsymbol{\theta},n}(t_{i-1})}{\lambda_{\boldsymbol{\theta}}(t_{i-1})} - \int_{t_{i-1}}^{t_i} \lambda'_{\boldsymbol{\theta},n}(x)dx\right) - \phi'\left(\int_0^{t_1} \lambda_{\boldsymbol{\theta}}(u)du - 1\right) \int_0^{t_1} \lambda'_{\boldsymbol{\theta},n}(x)dx
\end{pmatrix}^\top /T,
\end{aligned}$$

where $\lambda'_{\boldsymbol{\theta},k}(t)$ is the partial derivative of $\lambda_{\boldsymbol{\theta}}(t)$ with respect to θ_k for $k = 1, ..., n$.

Without loss of generality, we only need to consider the first entry of $\varrho(\boldsymbol{\theta})$. When $\boldsymbol{\theta}$ takes the true value $\boldsymbol{\theta}^*$, its gradient can be written as

$$\begin{aligned}
\varrho(\boldsymbol{\theta})_1 :=&\left(\sum_{i=1}^{I} \phi'\left(\int_{t_{i-1}}^{t_i} \lambda_{\boldsymbol{\theta}^*}(u)du - 1\right) \cdot \left(\frac{\lambda'_{\boldsymbol{\theta}^*,1}(t_{i-1})}{\lambda_{\boldsymbol{\theta}^*}(t_{i-1})} - \int_{t_{i-1}}^{t_i} \lambda'_{\boldsymbol{\theta}^*,1}(x)dx\right)\right. \\
&\left. - \phi'\left(\int_0^{t_1} \lambda_{\boldsymbol{\theta}^*}(u)du - 1\right) \int_0^{t_1} \lambda'_{\boldsymbol{\theta}^*,1}(x)dx\right)/T.
\end{aligned}$$

We define $\Lambda^*(t) := \int_0^t \lambda_{\boldsymbol{\theta}^*}(t)dt$. When $S = \{t_1, ..., t_I\}$ is a counting process with intensity function $\lambda_{\boldsymbol{\theta}^*}(t)$, we know that $\{\Lambda^*(t_1), \cdots, \Lambda^*(t_I)\}$ is a standard Poisson process on $\left[0, \int_0^T \lambda_{\boldsymbol{\theta}^*}(t)dt\right]$.

Then we deploy the time transformation technique, i.e., using $\Lambda^*(t)$ to replace t to discuss. Following the notation in Zhang et al., 2025, we define $\widetilde{\lambda_{\boldsymbol{\theta}^*}}(\Lambda^*(t)) = \lambda_{\boldsymbol{\theta}^*}(t)$ and $\widetilde{\kappa}(\Lambda^*(t)) = \lambda'_{\boldsymbol{\theta}^*,1}(t)$. We have

$$\begin{aligned}
&\mathbb{E}\left[\sum_{i=1}^{I} \phi'\left(\int_{t_{i-1}}^{t_i} \lambda_{\boldsymbol{\theta}^*}(u)du - 1\right) \cdot \left(\frac{\lambda'_{\boldsymbol{\theta}^*,1}(t_{i-1})}{\lambda_{\boldsymbol{\theta}^*}(t_{i-1})} - \int_{t_{i-1}}^{t_i} \lambda'_{\boldsymbol{\theta}^*,1}(x)dx\right) - \phi'\left(\int_0^{t_1} \lambda_{\boldsymbol{\theta}^*}(u)du - 1\right) \int_0^{t_1} \lambda'_{\boldsymbol{\theta}^*,1}(x)dx\right] \\
=&\underbrace{\mathbb{E}\left[\sum_{i=1}^{M} \phi'\left(\Lambda^*(t_i) - \Lambda^*(t_{i-1}) - 1\right) \cdot \left(\frac{\widetilde{\kappa}(\Lambda^*(t_{i-1}))}{\widetilde{\lambda_{\boldsymbol{\theta}^*}}(\Lambda^*(t_{i-1}))} - \int_{\Lambda^*(t_{i-1})}^{\Lambda^*(t_i)} \frac{\widetilde{\kappa}(\Lambda^*(t))}{\widetilde{\lambda_{\boldsymbol{\theta}^*}}(\Lambda^*(t))} d\Lambda^*(t)\right)\right]}_{\text{term1}} \\
&- \mathbb{E}\left[\phi'\left(\Lambda^*(t_1) - 1\right) \cdot \int_0^{\Lambda^*(t_1)} \frac{\widetilde{\kappa}(\Lambda^*(t))}{\widetilde{\lambda_{\boldsymbol{\theta}^*}}(\Lambda^*(t))} d\Lambda^*(t)\right].
\end{aligned}$$

For simplicity, we write $\widetilde{\kappa}(\Lambda^*(t))/\widetilde{\lambda_{\boldsymbol{\theta}^*}}(\Lambda^*(t))$ as $\chi(\Lambda^*(t))$, $\forall t \in [0, T]$, and $\left\{\widetilde{t}_1, \cdots, \widetilde{t}_I\right\} := \{\Lambda^*(t_1), \cdots, \Lambda^*(t_I)\}$. We define $\widetilde{N}(\widetilde{t}) = N(\Lambda^{*-1}(\widetilde{t}))$, then we know that $\mathbb{P}(\widetilde{N}(\widetilde{t} + d\widetilde{t}) - \widetilde{N}(\widetilde{t}) = 1) = \mathbb{P}(N(\Lambda^{*-1}(\widetilde{t} + d\widetilde{t})) - N(\Lambda^{*-1}(\widetilde{t})) = 1) = d\Lambda^*(\Lambda^{*-1}(\widetilde{t})) = d\widetilde{t}$. In other words, $\left\{\widetilde{t}_1, \cdots, \widetilde{t}_I\right\}$ becomes a standard homogeneous Poisson process after time transformation.

By calculations, we get that term1 can be simplified as

$$
\begin{aligned}
&\mathbb{E}\left[\sum_{i=1}^{I}\phi'\left(\Lambda^*(t_i)-\Lambda^*(t_{i-1})-1\right)\cdot\left(\chi(\Lambda^*(t_{i-1}))-\int_{\Lambda^*(t_{i-1})}^{\Lambda^*(t_i)}\chi(\Lambda^*(t))d\Lambda^*(t)\right)\right]\\
=&\mathbb{E}\left[\sum_{i=1}^{I}\phi'\left(\widetilde{t}_i-\widetilde{t}_{i-1}-1\right)\cdot\left(\chi(\widetilde{t}_{i-1})-\int_{\widetilde{t}_{i-1}}^{\widetilde{t}_i}\chi(\widetilde{s})d\widetilde{s}\right)\right] \qquad \text{(variable substitution)}\\
=&\mathbb{E}\left[\sum_{i=1}^{I}\mathbb{E}\left[\phi'\left(\widetilde{t}_i-\widetilde{t}_{i-1}-1\right)\cdot\left(\chi(\widetilde{t}_{i-1})-\int_{\widetilde{t}_{i-1}}^{\widetilde{t}_i}\chi(\widetilde{s})d\widetilde{s}\right)\mid\mathcal{H}_{\widetilde{t}_{i-1}}\right]\right] \qquad \text{(take the conditional expectation respectively)}\\
=&\mathbb{E}\left[\sum_{i=1}^{I}f(\widetilde{t}_{i-1})\right]=\mathbb{E}\left[\int f(\widetilde{t})d\widetilde{N}(\widetilde{t})\right]=\int_0^{\Lambda^*(T)}f(\widetilde{t})d\widetilde{t},
\end{aligned}
$$

where $f(\widetilde{t}_{i-1}):=\mathbb{E}\left[\phi'\left(\widetilde{t}_i-\widetilde{t}_{i-1}-1\right)\cdot\left(\chi(\widetilde{t}_{i-1})-\int_{\widetilde{t}_{i-1}}^{\widetilde{t}_i}\chi(\widetilde{s})d\widetilde{s}\right)\mid\mathcal{H}_{\widetilde{t}_{i-1}}\right]$ is a function of $\widetilde{t}_{i-1}$.

Because $\widetilde{t}_i-\widetilde{t}_{i-1}$ follows the standard exponential distribution conditioning on the history $\mathcal{H}_{\widetilde{t}}$, we can write $f(\widetilde{t})$ as $\mathbb{E}_{X\sim\text{EXP(1)}}\left[\phi'\left(X-1\right)\cdot\left(\chi(\widetilde{t})-\int_{\widetilde{t}}^{\widetilde{t}+X}\chi(\widetilde{s})d\widetilde{s}\right)\right]$.

We swap the order of integration of $\widetilde{t}$ and $\widetilde{s}$ and get

$$
\begin{aligned}
&\varrho(\boldsymbol{\theta})_1\\
=&\left(\int_0^{\Lambda^*(T)}\mathbb{E}_{X\sim\text{EXP(1)}}\left[\phi'\left(X-1\right)\cdot\left(\chi(\widetilde{t})-\int_{\widetilde{t}}^{\widetilde{t}+X}\chi(\widetilde{s})d\widetilde{s}\right)\right]d\widetilde{t}-\mathbb{E}\left[\phi'\left(\widetilde{t}_1-1\right)\cdot\int_0^{\widetilde{t}_1}\chi(\widetilde{s})d\widetilde{s}\right]\right)/T\\
=&\mathbb{E}_{X\sim\text{EXP(1)}}\left[\phi'\left(X-1\right)\left(\int_0^{\Lambda^*(T)}\chi(\widetilde{t})d\widetilde{t}-\int_0^X\int_0^{\Lambda^*(T)}\chi(\widetilde{t}+\widetilde{s})d\widetilde{t}d\widetilde{s}\right)\right]/T\\
&-\mathbb{E}_{X\sim\text{EXP(1)}}\left[\phi'\left(X-1\right)\int_0^X\chi(\widetilde{s})d\widetilde{s}\right]/T \qquad \text{(swap the order of integration of } \widetilde{t} \text{ and } \widetilde{s})\\
=&\int_0^{\cdot\Lambda^*(T)}\chi(\widetilde{t})d\widetilde{t}\cdot\mathbb{E}_{X\sim\text{EXP(1)}}\left[\phi'\left(X-1\right)\cdot\left(1-\int_0^X d\widetilde{s}\right)\right]/T \quad \text{(extract the part that is irrelevant to the expectation)}\\
&+\left(\mathbb{E}_{X\sim\text{EXP(1)}}\left[\phi'\left(X-1\right)\int_0^X\left(\int_0^{\widetilde{s}}\chi(u)du\right)d\widetilde{s}\right]-\mathbb{E}_{X\sim\text{EXP(1)}}\left[\phi'\left(X-1\right)\int_0^X\chi(\widetilde{s})d\widetilde{s}\right]\right)/T\\
=&0+C_g/T,
\end{aligned}
$$

where $C_g:=\mathbb{E}_{X\sim\text{EXP(1)}}\left[\phi'\left(X-1\right)\int_0^X\left(\int_0^{\widetilde{s}}\chi(u)du\right)d\widetilde{s}\right]-\mathbb{E}_{X\sim\text{EXP(1)}}\left[\phi'\left(X-1\right)\int_0^X\chi(\widetilde{s})d\widetilde{s}\right]$. Here we use the property that $\int_0^{\Lambda_k^*(T)}\chi(\widetilde{t})d\widetilde{t}=\int_0^{\Lambda_k^*(T)-\widetilde{s}}\chi(\widetilde{t}+\widetilde{s})d\widetilde{t}+\int_0^{\widetilde{s}}\chi(\widetilde{t})d\widetilde{t}$ and $\widetilde{t}_1$ follow the standard exponential distribution. By the assumption that $\nabla_{\boldsymbol{\theta}}\log\lambda_{\boldsymbol{\theta}}(t)$, $\lambda_\theta(t)$ are bounded at the true $\boldsymbol{\theta}^*$, hence C_g is bounded as well.

As a result, we get that $|\varrho(\boldsymbol{\theta})|$ is of order $O(1/T)$ when $\boldsymbol{\theta}=\boldsymbol{\theta}^*$. In other words, the gradient goes to 0 elementwisely as $T\to\infty$ when $\rho_1=\rho_2$.

B. Statistical Property

In the context of temporal point processes, the terms μ_1, μ_2, and μ_3 refer to the mean, variance, and covariance of the event counts within time intervals, respectively. These quantities provide crucial insights into the distribution of events and their temporal dependencies. Their precise definitions are as follows.

We partition the total time span $[0, T]$ into n time intervals, each of length τ. The number of events occurring within the j-th interval, denoted M_j, corresponds to the interval $[(j-1)\tau, j\tau)$.

μ_1 represents the empirical mean of the number of events within the time interval τ:

$$\mu_1 = \frac{1}{n}\sum_{j=1}^{n} M_j.$$

μ_2 denotes the empirical variance of the number of events within the same time interval:

$$\mu_2 = \frac{1}{n}\sum_{j=1}^{n}(M_j - \mu_1)^2.$$

μ_3 represents the empirical covariance of the number of events between two intervals separated by a lag Δ:

$$\mu_3 = \frac{1}{n_*}\sum_{j=1}^{n_*}(M_j \times M_{j+\Delta}) - \left(\frac{1}{n_*}\sum_{j=1}^{n_*} M_j\right)\left(\frac{1}{n_*}\sum_{j=1}^{n_*} M_{j+\Delta}\right),$$

where M_j denotes the number of events in the j-th interval, and $M_{j+\Delta}$ represents the number of events in the $j+\Delta$-th interval. Here, $n_* = n - \Delta$, with Δ representing the lag between the two intervals.

μ_3 plays a crucial role in capturing the self-excitation property of temporal point processes. Self-excitation refers to the phenomenon where past events increase the likelihood of future events, creating temporal correlations between events over time. A positive value of μ_3 suggests a correlation between the event counts in the j-th and $j+\Delta$-th intervals, indicating that events in one interval influence the occurrence of events in a subsequent interval. If μ_3 exhibits significant variation as Δ changes, it indicates the presence of self-excitation. A higher value of μ_3 for smaller lags suggests stronger temporal dependence (i.e., more pronounced self-excitation), while a constant or small μ_3 across varying Δ values may suggest weaker or absent self-excitation.

Table 5. The μ_3 values for different datasets at various lag values. For each dataset, the total time span is divided into $n = 30$ equal-length time intervals, with each interval's length given by $\tau = T/n$. The covariance of event counts within each time interval and the corresponding lag Δ is calculated for each sequence in the dataset. The average of these covariance values across all sequences is then used to obtain the μ_3 for the dataset.

DATASET	μ_3				
	Δ = 1	Δ = 2	Δ = 3	Δ = 5	Δ = 10
MIMIC-II	0.024	0.012	0.007	0.002	-0.003
RETWEET	7.220	4.554	3.216	1.753	0.361
EARTHQUAKE	0.644	0.433	0.306	0.152	0.023
STACKOVERFLOW	1.042	0.871	0.718	0.451	-0.091
AMAZOM	3.674	2.911	2.279	1.259	-0.010
VOLCANO	3.272	1.285	0.413	0.022	-0.138

Table 5 presents the values of μ_3 across different datasets for various lag values. The results indicate clear evidence of self-excitation in most cases, as the values of μ_3 consistently show a positive correlation between event counts in different time intervals. This correlation is particularly pronounced at smaller lag values, which strongly suggests the presence of self-excitation behavior. As the lag Δ increases, the values of μ_3 generally decrease, which aligns with the expected behavior of self-excitation, where the influence of past events on future events diminishes over time.

It is worth noting that the MIMIC-II dataset consists of relatively short sequences with fewer events, which limits the ability to effectively capture the covariance between event counts in different time intervals. In contrast, the other datasets, with a larger number of events, yield more consistent results in identifying self-excitation patterns.

C. Detailed Analysis of RED

C.1. Influence Function

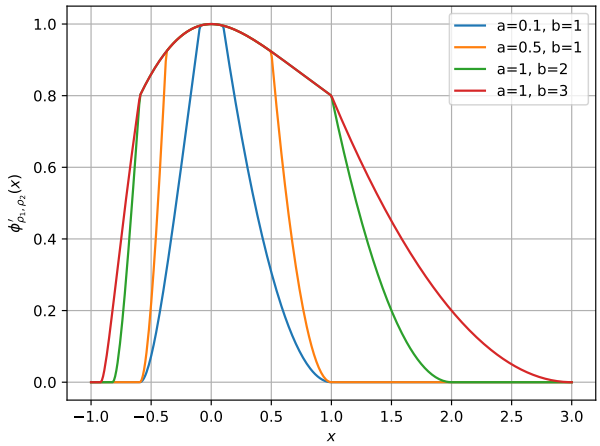

Figure 3. The influence function $\phi'_{\rho_1,\rho_2}(x)$ under different parameter settings (a, b), with $\rho_1 = \rho_2 = 1$. The parameters a and b determine the transition points and the degree of truncation applied to the function.

Figure 3 visualizes the behavior of the influence function $\phi'_{\rho_1,\rho_2}(x)$ defined in (3), with the parameters set as $\rho_1 = \rho_2 = 1$. The function exhibits piecewise behavior, where the varying parameters a and b define the transition points and the extent of the truncation. Specifically, $\phi'_{\rho_1,\rho_2}(x)$ remains relatively high for $x \in [0, a]$ and then decays quadratically in the interval $[a, b]$. Beyond b, $\phi'_{\rho_1,\rho_2}(x)$ becomes zero. The function attains its maximum at $x = 0$, and maintains a certain symmetry for negative and positive arguments due to the definition: $\phi'(x) = \phi'(x')$ for $-1 \leq x < 0$, where x' is defined as $x' := \{x' | (x' + 1)\exp(-x' - 1) = (x + 1)\exp(-x - 1), x' \geq 0\}$. The interplay between the parameters a and b alters the transition regions of the function. Smaller values of a and b impose stricter constraints on x, which, in turn, affects the sensitivity of the weight function to the underlying integration values.

This visualization demonstrates the flexibility of the influence function, as the parameters a, b, ρ_1 and ρ_2 allow for fine-tuning of the sensitivity and truncation behaviors, which are essential for robust statistical modeling. Specifically, the parameters a, b can be adjusted based on the characteristics of the data without affecting the order of the function values $\phi'_{\rho_1,\rho_2}(x)$, as shown by the following proposition.

Proposition C.1. *Let $\rho_1 = \rho_2 = 1$. For any $x_1, x_2 \in [-1, \infty)$, let $\phi'_{\rho_1,\rho_2}(x)$ be denoted as $\phi'_1(x)$ when the parameters are $a = a_1$ and $b = b_1$, and as $\phi'_2(x)$ when the parameters are $a = a_2$ and $b = b_2$, where $a_1, a_2, b_1, b_2 > 0$. It holds that:*

$$\phi'_1(x_1) \geq \phi'_1(x_2) \quad \Leftrightarrow \quad \phi'_2(x_1) \geq \phi'_2(x_2).$$

Proof. When $x_1, x_2 \in [-1, 0)$, both $\phi'_1(x)$ and $\phi'_2(x)$ are monotonically decreasing. Therefore, we have:

$$\phi'_1(x_1) \geq \phi'_1(x_2) \quad \Leftrightarrow \quad x_1 \leq x_2 \quad \Leftrightarrow \quad \phi'_2(x_1) \geq \phi'_2(x_2).$$

When $x_1, x_2 \in [0, \infty)$, both $\phi'_1(x)$ and $\phi'_2(x)$ are monotonically increasing. Hence, we get:

$$\phi'_1(x_1) \geq \phi'_1(x_2) \quad \Leftrightarrow \quad x_1 \geq x_2 \quad \Leftrightarrow \quad \phi'_2(x_1) \geq \phi'_2(x_2).$$

Without loss of generality, consider when $x_1 \in [-1, 0)$ and $x_2 \in [0, \infty)$, from the definition of $\phi'_{\rho_1,\rho_2}(x)$, we know that $\phi'_1(x_1) = \phi'(x_1) = \phi'(x'_1) = \phi'_1(x'_1)$, where x'_1 satisfies the equation $(x'_1 + 1)\exp(-x'_1 - 1) = (x_1 + 1)\exp(-x_1 - 1)$, with $x'_1 \geq 0$. Similarly, we have $\phi'_2(x_1) = \phi'_2(x'_1)$. Therefore, the inequality becomes:

$$\phi'_1(x_1) \geq \phi'_1(x_2) \quad \Leftrightarrow \quad \phi'_1(x'_1) \geq \phi'_1(x_2) \quad \Leftrightarrow \quad x'_1 \geq x_2 \quad \Leftrightarrow \quad \phi'_2(x'_1) \geq \phi'_2(x_2) \quad \Leftrightarrow \quad \phi'_2(x_1) \geq \phi'_2(x_2).$$

□

C.2. Weight Function

Based on the definition of the influence function, we calculate the weight value for each event using Equation (2). When the integral $\int_{t_{i-1}}^{t_i} \lambda^{(1)}(u)\, du$ is close to 1, $W_i(S;\boldsymbol{\theta})$ approaches 1. Conversely, $W_i(S;\boldsymbol{\theta})$ approaches 0 as the integral deviates from 1. This allows us to isolate the residual events.

In all experiments presented in this paper, we set $\rho_1 = \rho_2 = 1$. The parameters a and b are adjusted based on the characteristics of the data from different datasets to ensure that the proportion of residual events remains within an appropriate range. The variation of parameters a and b influences the values of the weights and the proportion of non-zero weights, but does not affect the ranking of the weight values, as demonstrated in Proposition C.1. This further justifies the use of the weight function we define to separate residual events.

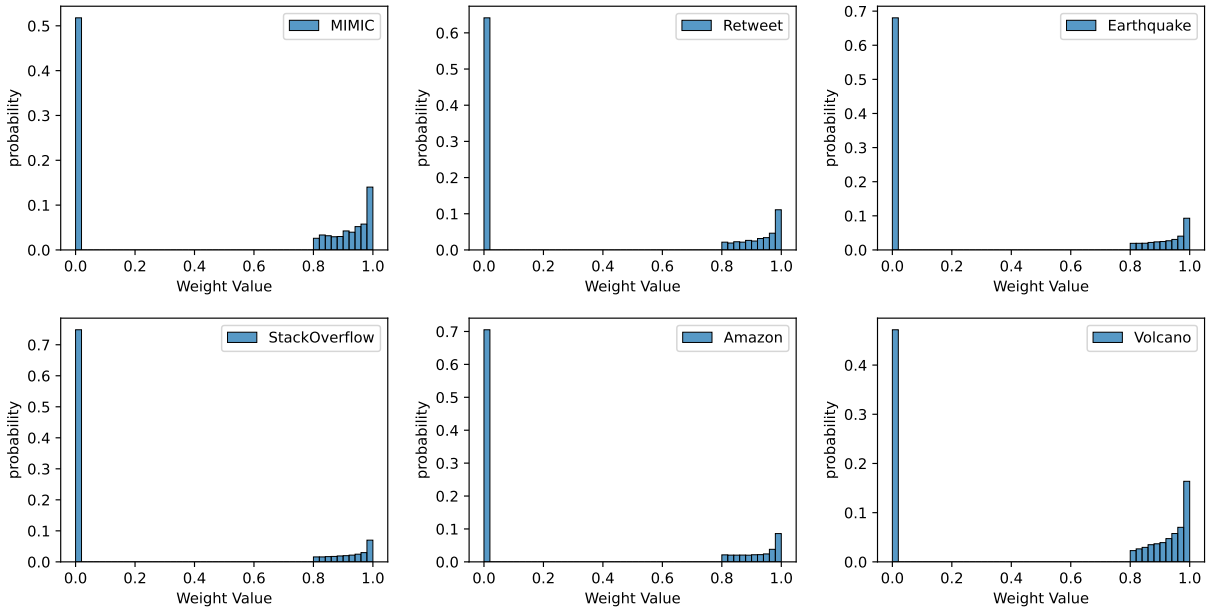

Figure 4. Weight distribution across different datasets: (a) MIMIC-II, (b) Retweet, (c) Earthquake, (d) StackOverflow, (e) Amazon, (f) Volcano. The weight computation is performed as follows: First, a Hawkes process with a fixed baseline intensity and an exponential decay function is fitted to the data to obtain the intensity function. The parameters of the influence function are set to $a = 1$, $b = 2$, $\rho_1 = \rho_2 = 1$ Based on this, the weight values for all events across all sequences in the dataset are computed. The histograms show the distribution of these weight values for each dataset.

Figure 4 illustrates the distribution of weight values computed across different datasets under identical settings. Each distribution exhibits a truncation near a weight value of 0.8, with a significant portion of the probability mass concentrated at 0. A higher weight value indicates that the number of events predicted by the Hawkes process closely matches the actual number of observed events, suggesting that the model accurately captures event occurrences within this interval. Conversely, a lower weight value indicates poorer performance of the intensity function.

Given this observation, it is natural to consider filtering the events based on their weights, which would enable us to isolate the residual component of the events. These residual events represent unexpected events that may not have clear statistical interpretations. Consequently, we can further utilize neural TPPs to model the intensity of this residual component. This filtering process could therefore be particularly useful for capturing statistical properties such as periodicity and self-excitation, while the residual component can be further analyzed using additional modeling techniques.

C.3. Combined Intensity

In our approach, the final intensity function for event type k is defined as a combination of the intensity functions from the Hawkes process and the neural TPP for residual events. Specifically, the combined intensity function is given by:

$$\lambda_k(t) = (1-\alpha)\lambda_k^{(1)}(t) + \lambda_k^{(2)}(t),$$

where $\alpha = \frac{|\mathbf{S}'_w|}{|\mathbf{S}|}$ represents the proportion of residual events in $\mathbf{S}'_w$ relative to the total number of events in the original dataset $\mathbf{S}$. This formulation leverages the superposition property of point processes, enabling the combination of contributions from both the Hawkes process and the neural TPP in a way that aligns with intuitive observations from the data. The weight parameter α is directly driven by the residual events present in the dataset, making the model's behavior data-driven and interpretable.

One might wonder the advantages of our proposed method in comparison to the approach of fitting the data directly using the intensity function of the form

$$\lambda_k(t) = (1-\alpha_1)\lambda_k^{(1)}(t) + \alpha_2\lambda_k^{(2)}(t), \tag{8}$$

where α_1 and α_2 are treated as learnable parameters. Here we give two explanations.

i. The latter approach (8) does not explicitly account for the proportion of residual events in the data; instead, the parameters α_1 and α_2 are optimized during training to minimize the loss function and determine the best-fit values. In contrast, our method explicitly identifies the residual events and non-residual events.

ii. The latter approach is much more computationally heavy. Fitting the model in form of (8) requires more computational time than that of fitting $\lambda_k^{(2)}(t)$ only. On the other hand, our method is plug-and-play module-based. Hence, Step 2 of the proposed algorithm effectively reduces the sample size and hence makes Step 3 more efficient. As a result, our method is more lightweight than the original approach, which fits the full data with $\lambda_k^{(2)}(t)$.

C.4. An Alternative Choice of Influence Function

The modular structure of the RED framework allows the substitution of $\phi'(x)$ with any valid influence function that satisfies the "unbiasedness" property. To further enrich this work, we investigate an alternative influence function given by

$$\phi'(x) = \frac{(1+\alpha)(x+1)}{(x+1)+\alpha\exp(x)}, \quad x \geq -1.$$

This function is continuously differentiable over its domain and preserves the "unbiasedness" property established in Proposition 3.1.

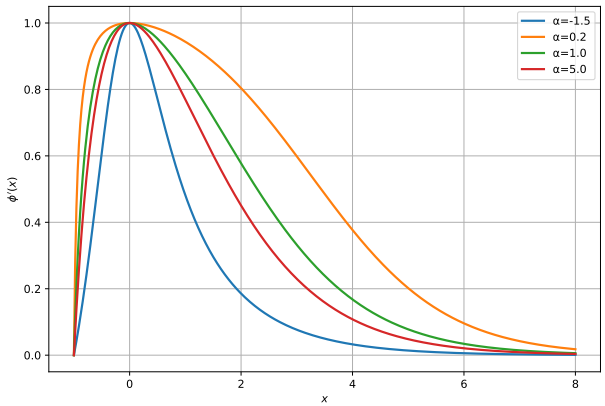

Figure 5. The smooth influence function $\phi'(x) = \frac{(1+\alpha)(x+1)}{(x+1)+\alpha\exp(x)}$, $x \geq -1$ under different parameter settings α.

As reported in Table 6, the Residual TPP method implemented with both the original and alternative influence functions (denoted as "old" and "new", respectively) consistently outperforms the baseline models, demonstrating the robustness of the RED framework to the choice of influence function.

Table 6. Performance Comparison on MIMIC-II and Earthquake Datasets. *Goodness-of-fit* (Log-Likelihood, higher is better); *Prediction* (Time RMSE / Type Error Rate, lower is better).

MODEL	MIMIC-II		EARTHQUAKE	
	LOG-LIKELIHOOD	TIME/TYPE	LOG-LIKELIHOOD	TIME/TYPE
MHP	-2.839	0.925/27.9%	-4.155	1.475/60.6%
RMTPP	-2.626	0.998/37.8%	-4.643	1.956/52.9%
RES RMTPP(OLD)	**-2.045**	**0.915**/26.7%	-3.689	1.420/**52.7%**
RES RMTPP(NEW)	-2.150	0.923/**24.4%**	**-3.667**	**1.415**/52.8%
NHP	-2.031	1.010/26.7%	-2.389	1.910/53.9%
RES NHP(OLD)	**-1.825**	0.913/20.9%	-1.930	**1.416/52.8%**
RES NHP(NEW)	-2.040	**0.900/18.0%**	**-1.916**	1.419/52.9%
SAHP	-4.672	0.971/23.8%	-3.338	1.463/54.1%
RES SAHP(OLD)	**-4.488**	0.935/**18.0%**	-3.335	**1.455**/53.3%
RES SAHP(NEW)	-4.571	**0.925**/22.1%	**-3.312**	1.458/**53.0%**
THP	-2.048	1.129/35.5%	-3.498	1.857/54.7%
RES THP(OLD)	**-2.040**	0.930/27.9%	**-3.415**	**1.403/52.8%**
RES THP(NEW)	-2.879	**0.911/26.7%**	-3.533	1.417/52.8%
ATTNHP	-2.500	1.030/35.4%	-2.896	1.822/54.5%
RES ATTNHP(OLD)	**-2.197**	0.932/36.0%	-3.147	**1.413/52.8%**
RES ATTNHP(NEW)	-2.918	**0.914/32.0%**	**-1.969**	1.419/52.8%
ODETPP	-1.855	1.416/22.1%	-2.203	2.396/56.0%
RES ODETPP(OLD)	**-1.371**	0.934/**19.2%**	-2.340	**1.412/53.0%**
RES ODETPP(NEW)	-2.365	**0.921**/19.3%	**-1.889**	**1.411**/53.1%

D. Experiment implementation details

D.1. Experimental Environment

All experiments in this study were implemented using PyTorch (Paszke et al., 2019), trained using the Adam (Kingma, 2014) optimizer, and executed on a CPU.

D.2. Descriptions of Datastes

- *MIMIC-II* (Johnson et al., 2016). This electronic medical dataset contains records of 650 patients' clinical visits to Intensive Care Units (ICUs) over a seven-year period. Each sequence represents the medical history of a single patient, where each event is defined by its timestamp and an associated disease diagnosis code, serving as the event type.
- *Retweet* (Zhou et al., 2013). The Retweet dataset includes 5,200 sequences of tweets, each consisting of an original tweet followed by a series of retweets. Each event in a sequence is characterized by its timestamp and the user tag of the individual retweeting the post. Furthermore, users are grouped into three categories based on their number of followers: "small", "medium", and "large," providing information as event type.
- *Earthquake* (Xue et al., 2024). This dataset comprises 4,296 sequences of timestamped earthquake events recorded across the Conterminous United States from 1996 to 2023. Each earthquake event is classified into one of seven categories based on its magnitude.
- *StackOverflow* (Leskovec & Krevl, 2014). This dataset includes two years of award collections from 2,200 users on StackOverflow, a question-and-answer website, where each user receives a sequence of badges. We treat each user's reward history as a sequence, with each event in the sequence representing the receipt of one of 22 distinct types of badges.
- *Amazon* (Ni, 2018). This dataset is derived from the product review behaviors of the 5,200 most active users on Amazon, spanning the period from January 2008 to October 2018. Each sequence represents a user's review history, where each event is defined by its timestamp and the category of the reviewed product. The dataset is particularly useful for studying consumer behavior and temporal dynamics in e-commerce.
- *Volcano* (Xue et al., 2024). This dataset consists of 431 sequences of timestamped volcanic eruption events recorded over several centuries across the globe. Since the dataset contains only a single event type, it represents a simpler scenario compared to other datasets. To address the extensive original time span and enhance model usability, we scale the time series by dividing the timestamps by one thousand, bringing the events into a more manageable time range for analysis.

Table 7. Statistics of the real-world datasets. The table columns, from left to right, represent the dataset, number of event types, number of events, sequence length, and number of sequences.

DATASET	# TYPES	# EVENTS	SEQUENCE LENGTH			# SEQUENCES		
			MIN	MEAN	MAX	TRAIN	VALID	TEST
MIMIC-II	75	2,419	2	4	33	527	58	65
RETWEET	3	493,708	10	40	97	9,000	1,535	1,520
EARTHQUAKE	7	70,723	11	16	18	3,000	400	896
STACKOVERFLOW	22	142,777	41	65	101	1,400	400	400
AMAZON	16	330,000	14	45	94	6,454	922	1,851
VOLCANO	1	9,127	1	14	244	400	50	181

D.3. Descriptions of Baseline Neural TPPs

We provide a detailed overview of the neural TPP models utilized in this study:

- **RMTPP** (Du et al., 2016): RMTPP utilizes Recurrent Neural Networks (RNNs) to capture the temporal dependencies in event sequences and learn a latent representation of event history. The core idea is to use an RNN to model the

Table 8. Statistics of the synthetic datasets. **Possion-based** dataset represented sequences follow periodic non-homogenous Possion process with homogeneous Poisson anomalies. **AttNHP-based** dataset represented TPP sequences are generated by AttNHP model and than randomly add homogeneous Poisson anomalies. **Possion+AttNHP** dataset represented sequences follow periodic non-homogenous Possion process with commission events generated by AttNHP. The table columns, from left to right, represent the dataset, number of event types, totoal number of events, number of residual events generated by $\lambda^{(2)}$, sequence length, and number of sequences.

DATASET	# TYPES	# EVENTS	# RESIDUALS	SEQUENCE LENGTH			# SEQUENCES		
				MIN	MEAN	MAX	TRAIN	VALID	TEST
POSSION-BASED	5	33,243	4,253	20	29	100	600	200	200
ATTNHP-BASED	5	100,052	12,807	9	44	79	1,000	500	500
POSSION+ATTNHP	5	54,737	16,625	24	55	126	600	200	200

sequence of past events, which then produces a dynamic, evolving feature vector that encodes the event history. This latent representation is then passed through an exponential transformation to define the intensity function.

- **NHP** (Mei & Eisner, 2017): NHP extends the traditional Hawkes process by introducing a continuous-time LSTM network to encode event sequences. Unlike standard Hawkes models, NHP allows the intensity function to decay over time automatically as a result of the learned parameters within the LSTM, without requiring inter-event times as inputs. This enables the model to learn complex temporal patterns without the need to manually specify the relationship between event timings.
- **SAHP** (Zhang et al., 2020a): SAHP introduces a self-attention mechanism to aggregate historical events for modeling the intensity function. This method enhances the expressiveness of the intensity function by allowing it to adapt to different time scales and dependencies. The self-attention mechanism enables the model to efficiently capture long-range dependencies in the event sequence.
- **THP** (Zuo et al., 2020): THP incorporates a Transformer architecture to model the intensity function of a temporal point process. The Transformer is known for its ability to capture long-range dependencies through self-attention mechanisms, which is particularly useful for event sequences with complex, non-linear temporal dependencies.
- **AttNHP** (Yang et al., 2022): AttNHP extends the Transformer framework to model event sequences, generating rich, context-sensitive embeddings for both observed and potential future events. These embeddings are based on the event's temporal context, enabling the model to generate more accurate predictions by considering the surrounding events and their relative importance.
- **ODETPP** (Xue et al., 2024):The hidden state evolution in the ODETPP is governed by a neural ODE. It is a simplified version of the Neural Spatio-Temporal Point Processes (NSTPP) proposed by Chen et al. (2021) by removing the spatial component. NSTPP leverages the Neural ODE framework to model high-fidelity spatio-temporal distributions, combining concepts from Neural Jump SDEs (Jia & Benson, 2019) and continuous-time normalizing flows (Chen et al., 2018).
- **FullyNN** (Omi et al., 2019): FullyNN is a model with a fully neural network-based intensity function. It uses a feedforward neural network to model the integral of the intensity function, deriving the intensity function as its derivative. This approach enables exact evaluation of the log-likelihood function, which involves the integral, without relying on numerical approximations.

D.4. Likelihood computation

We can estimate the model parameters by locally maximizing the Negative log-likelihood function with any stochastic gradient method:

$$\mathrm{NLL}(\lambda_k) = -\sum_{i=1}^{I} \log \lambda_{k_i}(t_i) + \sum_{k=1}^{K} \int_{t=0}^{T} \lambda_k(t)\, dt. \tag{9}$$

Note that computing the NLL can be challenging due to the presence of the integral in the second term of Equation (9). In this study, we utilize a Monte Carlo method for integral estimation to evaluate the intensity function and its gradient,

a technique first applied to the NLL computation for TPPs by Mei and Eisner (2017), and subsequently adopted by most models in the field (Omi et al., 2019; Xue et al., 2024).

We randomly sample time points $t \sim \text{Unif}(0, T)$ from the interval $[0, T]$ and compute the corresponding intensity $\lambda_k(t)$ for each event type k. The expected value of the integral is approximated by $T\lambda(t)$, and its gradient is computed via backpropagation, enabling efficient estimation of $\nabla\Lambda$. To enhance the accuracy of the integral and its gradient, we repeat this process over N samples, thereby reducing the variance of the noisy estimator.

E. More Experimental Results

E.1. Extending RED with Different Base Models

To further demonstrate the capability of the RED technique, we conduct additional experiments using two configurations: (i) simple TPP + RED + simple TPP, and (ii) neural TPP + RED + neural TPP. These configurations allow us to compare the performance of Residual TPP models that apply the RED technique with different base models for residual filtering, against their respective base models without RED.

As shown in Table 9, the combination of MHP + RED + MHP may yield worse results, as the model complexity of it is twice that of a single MHP. Apparently, the residuals do not follow MHP, leading to overfitting.

We then use the NHP as a representative neural base model for residual filtering. For each baseline neural TPP, we compare its performance with two RED variants: the original version using a Hawkes process as the base model and the new vairant using NHP. The results in Table 9 demonstrate that Residual TPPs with the RED technique consistently outperform the baselines, regardless of whether Hawkes or NHP is used as the base model. This highlights that the RED technique, as a plug-and-play module, can effectively enhance the performance of TPPs.

Table 9. Performance comparison of Residual TPPs with RED method using different base models on example benchmark datasets: MIMIC-II and Earthquake. For each baseline neural TPP, the first row reports its original performance, the second row (e.g. Res RMTPP) shows Residual TPP with RED using Hawkes as base model and the third row (e.g. NHP+RMTPP) presents the new Residual TPP with RED using NHP as base model. We evaluate the model's *goodness-of-fit* (Log-Likelihood, higher is better) and *prediction performance* (Time RMSE / Type Error Rate, lower is better).

Model	MIMIC-II		Earthquake	
	Log-Likelihood	Time/Type	Log-Likelihood	Time/Type
MHP	-2.839	0.925/27.9%	-4.155	1.475/60.6%
MHP+MHP	-3.883	0.919/43.0%	-3.746	1.452/71.8%
RMTPP	-2.626	0.998/37.8%	-4.643	1.956/52.9%
Res RMTPP	**-2.045**	**0.915/26.7%**	**-3.689**	**1.420/52.7%**
NHP+RMTPP	-2.059	0.929/37.8%	-4.098	1.797/52.8%
NHP	-2.031	1.010/26.7%	-2.389	1.910/53.9%
Res NHP	-1.825	**0.913/20.9%**	**-1.930**	**1.416/52.8%**
NHP+NHP	**-1.792**	1.061/22.1%	-2.089	1.711/52.8%
SAHP	-4.672	0.971/23.8%	-3.338	1.463/54.1%
Res SAHP	-4.488	**0.935/18.0%**	-3.335	1.455/53.3%
NHP+SAHP	**-4.126**	0.935/20.9%	**-3.152**	**1.452/52.8%**
THP	-2.048	1.129/35.5%	-3.498	1.857/54.7%
Res THP	**-2.040**	**0.930/27.9%**	**-3.415**	**1.403/52.8%**
NHP+THP	-2.743	1.067/35.5%	-4.112	1.755/53.0%
AttNHP	-2.500	1.030/35.4%	-2.896	1.822/54.5%
Res AttNHP	**-2.197**	**0.932**/36.0%	-3.147	**1.413/52.8%**
NHP+AttNHP	-2.649	1.094/**23.8%**	**-2.133**	1.757/52.8%
ODETPP	-1.855	1.416/22.1%	-2.203	2.396/56.0%
Res ODETPP	**-1.371**	**0.934/19.2%**	-2.340	**1.412/53.0%**
NHP+ODETPP	-2.480	1.263/22.7%	**-2.089**	2.099/54.8%

Nevertheless, we omit these extended results from the main text, as one of the key advantages of our method is its lightweight

nature. Our primary motivation is to capture statistical properties using a simple TPP and refine residuals using a neural TPP, thereby accelerating neural TPP computation with fewer events. While a neural TPP + RED + neural TPP configuration may further improve accuracy, it comes at the cost of substantially increased computational complexity.

E.2. FullyNN

To be self-complete, we compare the proposed method with FullyNN model. It can be clearly seen from Table 10 - Table 12 that there is a significant improvement in all performance metrics by using our RED technique.

Table 10. Performance of FullyNN and Residual FullyNN on the goodness-of-fit task. Higher scores indicate better performance.

MODEL	GOODNESS-OF-FIT (LOG-LIKELIHOOD)					
	MIMIC-II	RETWEET	EARTHQUAKE	STACKOVERFLOW	AMAZON	VOLCANO
FULLYNN	-10.01	-11.45	-10.00	-12.72	-9.737	-9.337
RES FULLYNN	**-2.971**	**-4.810**	**-4.132**	**-3.448**	**-1.963**	**-1.344**

Table 11. Performance of FullyNN and Residual FullyNN on the next event time prediction task. Lower scores indicate better performance.

MODEL	PREDICTION PERFORMANCE (TIME RMSE)					
	MIMIC-II	RETWEET	EARTHQUAKE	STACKOVERFLOW	AMAZON	VOLCANO
FULLYNN	4.580	21.92	4.313	4.316	3.862	6.267
RES FULLYNN	**1.247**	**19.76**	**1.523**	**2.379**	**1.196**	**5.629**

Table 12. Training time per epoch for FullyNN and Residual FullyNN. All training was conducted on a CPU.

MODEL	TRAINING TIME PER EPOCH (SECONDS)					
	MIMIC-II	RETWEET	EARTHQUAKE	STACKOVERFLOW	AMAZON	VOLCANO
FULLYNN	0.672	2.360	0.341	1.695	4.980	0.150
RES FULLYNN	**0.570**	**2.124**	**0.317**	**1.686**	**4.740**	**0.116**