# SMOOTH NEURAL POINT PROCESSES VIA B-SPLINES

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Temporal point processes (TPPs) provide a general and flexible framework for modeling sequences of events in continuous time. Neural networks have been successfully employed to model TPPs in a highly expressive and data-driven way. Neural TPPs are typically trained via Maximum Likelihood Estimation (MLE) by minimizing the negative log-likelihood (NLL), which depends on both the conditional intensity function (CIF) and its integral over time, the compensator. Recent neural TPP approaches enable exact evaluation of the NLL without numerical integration. However, these methods typically model the compensator rather than the CIF directly, impose constraints on the neural network architecture, and are computationally expensive during training, as event contributions to the NLL are evaluated sequentially rather than in parallel. In this work, we propose a novel neural TPP model that directly parametrizes the CIF as a non-negative combination of B-spline basis functions, whose coefficients are predicted by a neural network. This formulation enables exact evaluation of the NLL, preserves full flexibility in the neural architecture, allows efficient parallelization during training, and naturally supports CIF smoothness regularization through the integrated squared second derivative. Experiments on both synthetic and real-world datasets show improved computational efficiency and predictive accuracy compared to modern neural TPP baselines.

## 1 INTRODUCTION

Temporal point processes (TPPs) (Daley et al., 2003) provide a general and flexible framework for modeling sequences of events in continuous time. Event sequences are common in many disciplines, such as seismology (Ogata, 1999), biology (Truccolo et al., 2005), epidemiology (Diggle, 2013), social sciences (Mohler et al., 2011), and finance (Bacry et al., 2015; Hawkes, 2018).

A central object in the theory of TPPs is the conditional intensity function (CIF), which describes the instantaneous expected rate of events given the past history.

Classical models such as Poisson and Hawkes processes (Hawkes, 1971; Laub et al., 2021) provide parametric forms for the CIF. These models are attractive due to their interpretability and tractability, but they may be too restrictive to capture complex temporal dependencies observed in real-world event streams.

A straightforward extension of classical parametric models is to adopt a black-box approach and rely on neural networks to learn complex temporal dependencies directly from data. Despite its flexibility, this approach presents important computational challenges. Neural TPPs are typically trained via Maximum Likelihood Estimation (MLE) by minimizing the negative log-likelihood (NLL), which depends on both the CIF and its integral over time, the compensator. While evaluating the CIF is straightforward, computing the integral term can be difficult when the intensity is represented by a neural network. Several neural TPP models, including the Neural Hawkes Process (NHP) (Mei & Eisner, 2017), the Attentive Neural Hawkes Process (AttNHP) (Mei et al., 2021), and the State-Space Point Process (S2P2) (Chang et al., 2026), approximate the integral term numerically, typically through Monte Carlo estimation. However, this solution may increase the computational cost and introduce instability during training.

An alternative approach was introduced by Omi et al. (2019). The authors used their Fully Neural Network (FullyNN) to model the compensator instead of the CIF, and then recover the CIF via automatic differentiation (Baydin et al., 2018). While this approach effectively avoids numerical



Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

integration, it requires enforcing monotonicity of the compensator through architectural constraints (Sill, 1997; Chilinski & Silva, 2020). These constraints limit the flexibility of the model and, importantly, prevent parallel evaluation, since the intensity of every time point is computed sequentially. As a result, training and inference are slow, making the approach infeasible in data-intensive settings.

In this work, we introduce SmoothTPP, a novel neural TPP model that directly parametrizes the CIF using a non-negative combination of B-spline basis functions, whose coefficients are predicted by a neural network. This formulation enables exact and efficient evaluation of the NLL, eliminating the need for numerical integration. At the same time, it preserves full flexibility in the choice of the neural architecture, as no structural constraints are required to ensure validity of the model. Importantly, the intensity of the time points can be evaluated in parallel over time, leading to significantly improved computational efficiency compared to existing approaches. Finally, the proposed representation naturally enables smoothness regularization of the CIF via penalization of the integrated squared second derivative, providing additional control over the learned dynamics and better generalization.

## 2 Background

### 2.1 Temporal Point Processes (TPPs)

Intuitively, we can think of a point process $(T_n)_{n\geq 1}$ as an increasing sequence of random times

$$0 < T_1 < T_2 < \dots \ .$$

These random variables represent the occurrence (or arrival) times of events.

A point process can be associated with a counting process $N_t$, a stochastic process that counts the number of arrivals up to time $t$

$$N_t = \sum_{n\geq 1} \mathbf{1}_{\{T_n \leq t\}} \ .$$

An important quantity for a point process is the conditional intensity function (CIF), which represents the instantaneous rate of occurrence of events at time $t$, given the history up to time $t$ ($t$ excluded):

$$\lambda(t \mid \mathcal{F}_t) = \lim_{\Delta t \to 0^+} \frac{\mathbb{E}(N_{t+\Delta t} - N_t \mid \mathcal{F}_t)}{\Delta t} \ .$$

$\mathcal{F}_t$ is the filtration of the process, which represents the information available up to time $t$. In the following, we will use the simplified notation $\lambda(t) := \lambda(t \mid \mathcal{F}_t)$, that omits the explicit dependence of the CIF on the filtration.

### 2.2 Maximum Likelihood Estimation (MLE)

The most general approach for fitting the parameters of a point process is Maximum Likelihood Estimation (MLE), which consists in maximizing the likelihood of the observed sequence of events under the assumed model. The general form of the likelihood for point processes on the time period $[0, T]$ is

$$L = \left[\prod_{i=1}^{N_T} \lambda(T_i)\right] e^{-\Lambda(T)}$$

where $\Lambda(T) = \int_0^T \lambda(s)\, ds$ is the compensator of the process.

For numerical reasons, it is usually preferable to maximize the log-likelihood

$$\ell = \left(\sum_{i=1}^{N_T} \ln(\lambda(T_i))\right) - \Lambda(T)$$

or, alternatively, to minimize the negative log-likelihood (NLL), i.e., the log-likelihood with the sign reversed.

Often, it is more convenient to work with inter-event times $\tau_i = T_i - T_{i-1}$ instead of absolute times $T_i$. In this case, the log-likelihood can be written as

$$\ell = \sum_{i=1}^{N_T} \left( \ln(\phi(\tau_i)) - \Phi(\tau_i) \right)$$

where $\phi(\tau_i) = \lambda(T_i)$ is the CIF reparameterized in terms of the time elapsed since the last event, and $\Phi(\tau_i) = \int_0^{\tau_i} \phi(s)\, ds$ is the relative integral.

### 2.3 Random Time Change Theorem

The Random Time Change Theorem states that, given an increasing sequence of time points $\{T_1, T_2, \dots\}$ and a TPP with compensator $\Lambda(\cdot)$, the transformed sequence $\{\Lambda(T_1), \Lambda(T_2), \dots\}$ is a realisation of a unit rate Poisson process if and only if the original sequence $\{T_1, T_2, \dots\}$ is a realisation of the TPP defined by $\Lambda(\cdot)$.

By the Random Time Change Theorem, when the observed event times of a TPP are transformed through its compensator, the resulting sequence follows a unit-rate Poisson process. A key property of a unit-rate Poisson process is that its inter-arrival times are independent and exponentially distributed with rate 1, i.e., $\Phi(\tau_i) = \Lambda(T_i) - \Lambda(T_{i-1}) \sim \mathrm{Exp}(1)$, with $\Phi(\tau_i)$ being the cumulative integral of the CIF reparameterized in terms of the time elapsed since the last event.

Since an $\mathrm{Exp}(1)$ distribution has median $\log(2)$, a median-based estimator $\hat{\tau}_{i+1}$ for the waiting time until the next event can be obtained by finding $\hat{\tau}_{i+1}$ such that

$$\Phi(\hat{\tau}_{i+1}) = \log(2)\ . \tag{1}$$

### 2.4 B-splines

Splines are piecewise polynomial functions defined on a partition of an interval (Wood, 2017). A spline of degree $p$ with knots $\{x_j\}$ is a function that is a polynomial of degree $p$ on each subinterval $[x_j, x_{j+1})$ and has continuous derivatives up to order $p-1$ at the knots.

Spline spaces are dense in a wide class of function spaces, and therefore provide flexible approximations of target functions. In particular, by increasing the number of knots, spline functions can approximate arbitrarily well any target function under mild regularity conditions.

A convenient way to represent splines is through a B-spline basis (Green & Silverman, 1994). Any spline function $s(x)$ can be expressed as

$$s(x) = \sum_{k=1}^{K} w_k B_k(x)$$

where $w_k \in \mathbb{R}$ are coefficients and $\{B_k(x)\}_{k=1}^{K}$ are the non-negative B-spline basis functions. The B-spline basis has local support, meaning that each basis function is nonzero only on a limited number of adjacent knot intervals, leading to sparse and numerically stable representations. Moreover, since B-spline basis functions are non-negative, imposing non-negativity constraints on the coefficients $w_k \geq 0$ is sufficient to ensure that the resulting spline $s(x)$ is non-negative for all $x$.

Spline functions of degree $p = 3$, known as cubic splines, play a particularly important role. It can be shown that, among all functions defined on an interval $[a, b]$ that interpolate a set of points $\{(x_i, y_i)\}$, cubic splines are the smoothest in the sense of minimizing the roughness

$$\mathcal{P}(s) = \int_a^b (s''(x))^2 dx. \tag{2}$$

In the context of nonparametric regression, cubic splines arise as the exact solution to the functional optimization problem

$$\min_s \sum_{i=1}^{n} \left( y_i - s(x_i) \right)^2 + \alpha \mathcal{P}(s) \tag{3}$$

where $\alpha > 0$ controls the strength of the regularization.

The roughness penalty $\mathcal{P}(s)$ can be computed very efficiently. Indeed, it admits a quadratic form

$$\mathcal{P}(s) = \mathbf{w}^\top \mathbf{R} \mathbf{w}, \tag{4}$$

where $\mathbf{w}$ is the vector of spline coefficients and $\mathbf{R}$ is a positive semidefinite matrix with entries $R_{ij} = \int_a^b B_i''(x) B_j''(x)\, dx$ that depends only on the chosen basis. Importantly, $\mathbf{R}$ can be computed once in advance, and its structure is typically sparse due to the local support of B-spline basis functions, enabling efficient evaluation of the penalty during optimization.

# 3 METHODOLOGY

## 3.1 PROPOSED MODEL

We propose to model the CIF $\phi(\tau)$ on the inter-arrival times $\tau_i$ as a (cubic) B-spline function

$$\phi(\tau) = \sum_{k=1}^{K} w_k B_k(\tau).$$

The coefficients $w_k$ are predicted by a generic neural network that takes as input the history of past events and possibly additional contextual information. Since B-spline basis functions are non-negative, non-negativity of the CIF is guaranteed by construction by imposing $w_k \geq 0$ through suitable output activations of the neural network, for example softplus.

A key advantage of this parameterization is that the integral of the CIF, required for the NLL evaluation, can be computed in closed form as

$$\Phi(\tau) = \int_0^\tau \phi(s)\, ds = \sum_{k=1}^{K} w_k I_k(\tau),$$

where $I_k(\tau)$ are the integrated basis functions, which can be precomputed exactly before training.

Knots $\{k_j\}$ can be distributed uniformly over the domain of inter-arrival times, or chosen according to empirical quantiles of the observed inter-arrival time distribution. To avoid oscillations in the learned CIF, one can either use a limited number of knots $K$, or increase $K$ and add the roughness penalty $\mathcal{P}$ introduced in Equation 2 to the NLL in the loss function

$$-\sum_{i=1}^{N_T} \left(\ln(\phi(\tau_i)) - \Phi(\tau_i)\right) + \alpha \mathcal{P}(\phi). \tag{5}$$

While no optimality result like the regression problem in Equation 3 holds in the context of TPPs, penalizing the roughness of the CIF provides a principled way to control local fluctuations. Moreover, Equation 4 for the efficient computation of the penalization makes its inclusion during training essentially cost-free.

## 3.2 MULTIVARIATE EXTENSION

The proposed formulation naturally extends to multivariate TPPs with multiple event types $m \in \{1, \ldots, M\}$, each associated with its own conditional intensity function $\lambda_m(t)$.

In this setting, each conditional intensity function is modeled as

$$\phi_m(\tau) = \sum_{k=1}^{K_m} w_{m,k}\, B_{m,k}(\tau),$$

where $\{B_{m,k}(\tau)\}_{k=1}^{K_m}$ denotes the B-spline basis associated with event type $m$, and $w_{m,k} > 0$ are the corresponding coefficients predicted by the neural network.

The extension is immediate, as it only requires increasing the dimensionality of the network output to match the total number $\sum_{m=1}^{M} K_m$ of spline coefficients across all event types.

The roughness penalty naturally generalizes as a weighted sum over event types

$$P = \sum_{m=1}^{M} \alpha_m \, \mathbf{w}_m^\top \mathbf{R}_m \, \mathbf{w}_m,$$

where $\mathbf{w}_m = (w_{m,1}, \ldots, w_{m,K_m})^\top$ and $\mathbf{R}_m$ is the precomputed roughness matrix associated with the basis of type $m$. The coefficients $\alpha_m \geq 0$ control the strength of the smoothness regularization for each event type, allowing different levels of roughness penalization across the corresponding intensity functions.

For simplicity, one can also choose to use a shared temporal basis across all event types, instead of defining a distinct spline basis with different knot placements and numbers for each type.

### 3.3 Goodness of fit

Following common practice in the neural TPP literature, we evaluate the goodness of fit of the proposed model on the test data using two metrics:

- the average NLL per event;
- the mean absolute error (MAE) of the median-based estimator $\hat{\tau}_{i+1}$ for the next inter-arrival time described in Section 2.3 with respect to the true observed $\tau_{i+1}$.

Our model allows to obtain the estimator $\hat{\tau}_{i+1}$ in a very efficient way. Having an analytical expression for the compensator enables the median estimator to be obtained by directly solving Equation 1, without relying on Monte Carlo sampling as in intensity-based models such as NHP, AttNHP, and S2P2. Moreover, unlike FullyNN, a single forward pass of SmoothTPP provides the entire future trajectories of both the CIF and the compensator. Equation 1 can therefore be solved efficiently, as no additional forward passes are required during the iterations of the root-finding algorithm. Given the monotonicity of the compensator, we use the bisection method.

## 4 Experiments

We evaluate the proposed SmoothTPP model on 7 synthetic and 2 real-world datasets and compare it against four representative neural TPP baselines: NHP, FullyNN, AttNHP, and S2P2.

### 4.1 Datasets

Each synthetic dataset consists of a sequence of 100,000 time points, generated by different TPP models. Every sequence is then split into train and test sets, using a standard 80%/20% division. The TPPs considered are:

- a stationary Poisson process with $\lambda(t) = 1$;
- a non-stationary Poisson process with $\lambda(t) = 0.99 \cdot sin(\frac{2\pi t}{20000}) + 1$;
- a stationary Renewal process, in which the inter-event intervals $\tau_i$ are independent and identically distributed according to a log-normal probability distribution with mean $\mu = 1.0$ and standard deviation $\sigma = 6.0$;
- a non-stationary Renewal process, whose time points $T_i$ are obtained by first generating independent and identically distributed inter-event times $\tau_i'$ from a stationary Renewal process (Gamma distribution with mean $\mu = 1.0$ and standard deviation $\sigma = 0.5$), and then rescaling the time by inverting the formula $\tau_i' = \int_{T_{i-1}}^{T_i} r(s)\, ds$, where $r(t)$ is a non-negative trend function defined as $r(t) = 0.99 \, \sin\left(\frac{2\pi t}{20000}\right) + 1$;
- a Self-correcting process, with $\lambda(t) = \exp(t - \sum_{t_i < t} 1)$;
- a Hawkes process with single exponential kernel $\lambda(t) = \mu + \sum_{T_i < t} \alpha \, \beta \, \exp\left\{-\beta(t - T_i)\right\}$ and parameters $\mu = 0.2$, $\alpha = 0.8$, $\beta = 1$;

- a Hawkes process with multiple exponential kernel

$$\lambda(t) = \mu + \sum_{T_i<t} \sum_{j=1}^{M} \alpha_j \, \beta_j \, \exp\Big\{ -\beta_j (t - T_i) \Big\}$$

and parameters $M = 2$, $\mu = 0.2$, $\alpha_1 = \alpha_2 = 0.4$, $\beta_1 = 1.0$, $\beta_2 = 20.0$.

We additionally use the two publicly available real-world datasets employed in Omi et al. (2019), applying the same preprocessing, and measuring time in hours.

- Music dataset Celma (2010) contains last.fm users' listening history in January 2009. 100 sequences from the 100 most active users are created, using the first 80% of events in the sequences for training and the last 20% for testing.
- Meme dataset Leskovec et al. (2009) contains popular phrases collected from several online resources. 50 sequences from the 50 most used phrases during August 2008 are created, using the first 40 sequences for training and the last 10 sequences for testing.

### 4.2 Model architecture and training procedure

For SmoothTPP, we use a minimal Transformer architecture with embedding size 64, 4 attention heads, and a single vanilla decoder block as described in Vaswani et al. (2017), with an inner feed-forward layer dimension of 64. The input embedding is constructed as the sum of two components: a sinusoidal positional encoding to represent the chronological order of past events, and a learned embedding of the inter-arrival times obtained through a linear dense layer.

The CIF curve over future times is predicted by taking as input the inter-arrival times of the last 20 events. The CIF is parametrized as a cubic B-spline function with 20 internal knots distributed at quantiles of inter-arrival training times. The network outputs the positive coefficients of the B-splines using softplus activation in the last layer.

For all benchmark models, we use default or commonly adopted hyperparameter values, ensuring to obtain a number of trainable parameters comparable to our architecture. Specifically, we use a hidden size of 64 for NHP; an RNN encoder with hidden size 64, one recurrent layer, and two MLP layers of size 64 for FullyNN; a hidden size of 32, time-embedding size of 32, one layer, and four attention heads for AttNHP; and a hidden size of 64, four layers, and $P = 16$ for S2P2.

For all models, we used the same training protocol: Adam optimizer with learning rate 0.001, batch size 256, and maximum training epochs 200. Early stopping is applied based on the validation set, obtained as 20% of the training data, with patience 10 epochs, the NLL without penalization as monitored validation loss, and restoring the best weights at the end of the training.

For the SmoothTPP model, the validation set is also used to select for each dataset the optimal value of the regularization parameter $\alpha$, evaluated over a range of values from $10^{-3}$ to $10^{-10}$, also including the case with no regularization $\alpha = 0$.

SmoothTPP implementation relies on `Keras` Chollet et al. (2015) and `KerasHub` Watson et al. (2024) libraries, while we employed the implementation provided by EasyTPP library Xue et al. (2024) for all the other neural TPP models. Experiments were run on a T4 GPU provided by the Google Colab environment.

### 4.3 Computational performance evaluation

To ensure a fair comparison with the other benchmarks, the predictive accuracy results are obtained under the same training setting, using the cut sequence of last inter-event times to predict only the CIF at the next event.

However, our spline-based neural TPP model allows parallel evaluation of the CIF over multiple future times. In particular, given a sequence of events, our model is able to return the CIF evaluated at the time of each event in a single forward pass. This significantly reduces the training and testing time, and is especially beneficial when using a Transformer architecture.

We compare the training computational cost of the CIF evaluation scheme used in Omi et al. (2019) with the parallel multi-evaluation variant of our model. In the latter setting, the model processes

longer sequences (in our experiments, of length 100) and directly outputs the CIF at each event time in a single forward pass.

### 4.4 RESULTS

Tables 1, 2 show respectively the test NLL and MAE results of the experiments described in Section 4.

Table 1: Negative log-likelihood (NLL) on synthetic and real-world datasets. Best results are highlighted in bold.

| Type | Dataset | NHP | FullyNN | AttNHP | S2P2 | SmoothTPP |
|---|---|---|---|---|---|---|
| Synthetic | Stationary Poisson | **0.993** | 0.998 | **0.993** | 1.002 | 0.994 |
| | Non-stationary Poisson | 0.718 | 0.730 | **0.712** | 0.725 | 0.713 |
| | Stationary Renewal | 0.316 | 0.276 | 0.335 | **0.241** | 0.276 |
| | Non-stationary Renewal | **0.361** | 0.372 | 0.373 | 0.362 | 0.362 |
| | Self-correcting | 0.787 | 0.796 | 0.787 | 0.805 | **0.786** |
| | Hawkes 1 | **0.480** | 0.496 | 0.496 | 0.483 | 0.489 |
| | Hawkes 2 | -0.033 | 0.004 | 0.105 | **-0.040** | -0.006 |
| Real | Music | -2.122 | -2.132 | -1.435 | **-2.224** | -2.155 |
| | Meme | -1.964 | -1.671 | -1.332 | -2.560 | **-2.712** |

Table 2: Next-event time prediction MAE on synthetic and real-world datasets. Best results are highlighted in bold.

| Type | Dataset | NHP | FullyNN | AttNHP | S2P2 | SmoothTPP |
|---|---|---|---|---|---|---|
| Synthetic | Stationary Poisson | 0.776 | 0.761 | 0.761 | 0.792 | **0.688** |
| | Non-stationary Poisson | 0.786 | 0.778 | 0.760 | 0.898 | **0.708** |
| | Stationary Renewal | 1.002 | 1.009 | 1.018 | 1.050 | **0.975** |
| | Non-stationary Renewal | 0.455 | 0.475 | 0.577 | 0.477 | **0.409** |
| | Self-correcting | 0.566 | 0.584 | 0.541 | 0.594 | **0.495** |
| | Hawkes 1 | 0.954 | 0.956 | 0.906 | 0.957 | **0.888** |
| | Hawkes 2 | 1.093 | 1.021 | 1.009 | 1.090 | **0.987** |
| Real | Music | 0.225 | 0.207 | 0.194 | 0.211 | **0.183** |
| | Meme | 0.147 | 0.230 | 0.177 | 0.155 | **0.135** |

SmoothTPP achieves the lowest MAE on all seven synthetic datasets and both real-world datasets, consistently outperforming the considered neural TPP baselines in next-event time prediction. The NLL results are more heterogeneous: SmoothTPP achieves the best performance on the Self-correcting and Meme datasets and remains competitive with the best-performing baseline on the remaining datasets.

In most experiments, the optimal smoothing parameter $\alpha$ for SmoothTPP was nonzero, suggesting that regularization was indeed beneficial. The effect of regularization on the learned CIF from non-stationary Renewal and Hawkes 2 processes is shown in Figures 1 and 2.

A higher value of the regularization parameter $\alpha$ reduces the fluctuations of the predicted CIF, as is evident in the non-stationary Renewal process in Figure 1. In both cases, values of $\alpha$ not far from the optimal one do not lead to a significant degradation in performance, indicating that the model is reasonably robust to this parameter.

Using the T4 GPU provided by the Google Colab environment, one epoch with the standard training setting takes on average 1750 ms on synthetic datasets, whereas with our parallel multi-evaluation approach described in Section 4.3, an epoch takes on average 150 ms, corresponding to a speed-up of approximately 12×.

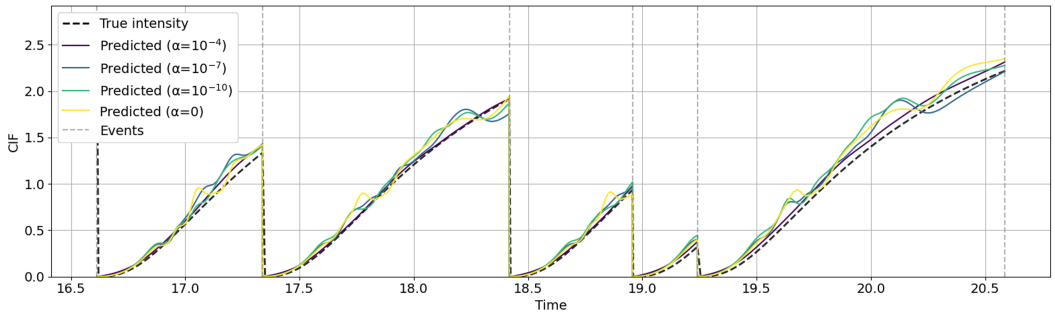



Figure 1: Regularization effect of different smoothing parameters $\alpha$ on the learned CIF from the non-stationary Renewal process.

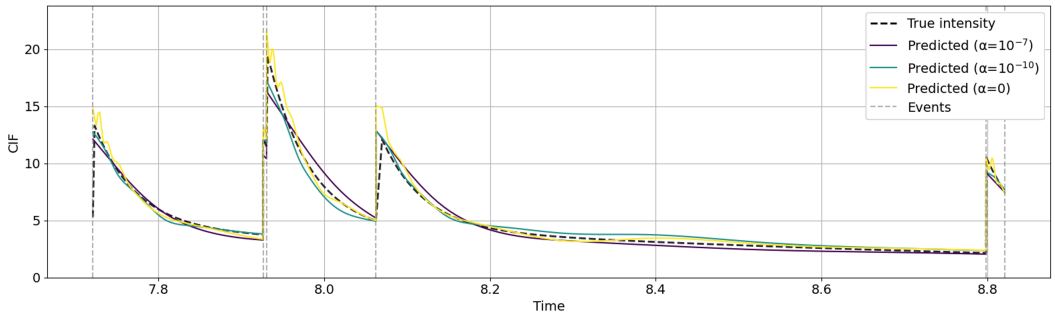



Figure 2: Regularization effect of different smoothing parameters $\alpha$ on the learned CIF from the Hawkes 2 process.

## 5 Discussion

In this work, we introduced a novel neural TPP model that directly parametrizes the CIF using a non-negative combination of B-spline basis functions, whose coefficients are predicted by a generic neural network. This method preserves exact evaluation of the NLL, but, unlike previous approaches, it allows full flexibility in the choice of the neural architecture and, thanks to the extensive parallelization, enables fast training on large datasets. Moreover, the smoothing regularization through the integrated squared second derivative prevents the oscillatory behavior typically associated with splines and leads to improved generalization.

We evaluated the proposed approach on both synthetic and real-world datasets and compared with other popular modern neural TPP models. Results show consistently improved next-event time prediction, with SmoothTPP achieving the lowest MAE on all the datasets considered. In terms of NLL, SmoothTPP remains competitive with the best-performing baseline across the considered datasets.

Beyond the improvement in predictive performance, the main strength of SmoothTPP lies in combining analytical tractability with massive parallelization and efficient computational cost. In our experiments, this combination resulted in an approximately $12\times$ training speed-up. Importantly, this computational advantage is achieved without sacrificing predictive performance or constraining the choice of the neural architecture. This opens the possibility of applying exact-likelihood neural TPPs to substantially larger datasets and longer event sequences, where the computational cost of existing approaches can become a limiting factor.

### AI use statement

Generative AI tools were used to improve the clarity and readability of the manuscript and to assist with the implementation of limited portions of the code. All AI-assisted text was reviewed by the

authors, and AI-assisted code was reviewed and tested for correctness. We take full responsibility for the final content of this work, including all AI-assisted text and code.

### ETHICS STATEMENT

This work uses synthetic data and publicly available datasets and does not involve the collection of new data from human subjects. We are not aware of any specific ethical concerns, privacy risks, or potential harmful applications arising from the research presented in this work.

### REPRODUCIBILITY STATEMENT

The experimental setup, model architecture, training procedure, and hyperparameter selection are described in the main paper. The source code required to reproduce the experiments is provided as supplementary material, together with a README containing instructions for its use. Upon acceptance, the code will be made publicly available in an online repository.

## REFERENCES

Emmanuel Bacry, Iacopo Mastromatteo, and Jean-François Muzy. Hawkes processes in finance. *Market Microstructure and Liquidity*, 1(01):1550005, 2015.

Atilim Gunes Baydin, Barak A Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. Automatic differentiation in machine learning: a survey. *Journal of machine learning research*, 18(153):1–43, 2018.

Oscar Celma. Last.fm dataset. `http://ocelma.net/MusicRecommendationDataset/lastfm-360K.html`, 2010.

Yuxin Chang, Alex Boyd, Cao Danica Xiao, Taha Kass-Hout, Parminder Bhatia, Padhraic Smyth, et al. Deep continuous-time state-space models for marked event sequences. *Advances in neural information processing systems*, 38:162672–162713, 2026.

Pawel Chilinski and Ricardo Silva. Neural likelihoods via cumulative distribution functions. In *Conference on uncertainty in artificial intelligence*, pp. 420–429. PMLR, 2020.

François Chollet et al. Keras. `https://keras.io`, 2015.

Daryl J Daley, David Vere-Jones, et al. *An introduction to the theory of point processes: volume I: elementary theory and methods*. Springer, 2003.

Peter J Diggle. *Statistical analysis of spatial and spatio-temporal point patterns*. CRC press, 2013.

Peter J. Green and Bernard W. Silverman. *Nonparametric Regression and Generalized Linear Models*. Springer US, Boston, MA, 1994. ISBN 978-0-412-30040-0 978-1-4899-4473-3. doi: 10.1007/978-1-4899-4473-3. URL `http://link.springer.com/10.1007/978-1-4899-4473-3`.

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

Alan G Hawkes. Hawkes processes and their applications to finance: a review. *Quantitative Finance*, 18(2):193–198, 2018.

Patrick J Laub, Young Lee, and Thomas Taimre. *The elements of Hawkes processes*. Springer, 2021.

Jure Leskovec, Lars Backstrom, and Jon Kleinberg. Meme-tracking and the dynamics of the news cycle. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '09)*, pp. 497–506. ACM, 2009. `https://snap.stanford.edu/data/memetracker9.html`.

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

Hongyuan Mei, Chenghao Yang, and Jason Eisner. Transformer embeddings of irregularly spaced events and their participants. In *International conference on learning representations*, 2021.

George O Mohler, Martin B Short, P Jeffrey Brantingham, Frederic Paik Schoenberg, and George E Tita. Self-exciting point process modeling of crime. *Journal of the american statistical association*, 106(493):100–108, 2011.

Yosihiko Ogata. Seismicity analysis through point-process modeling: A review. *Seismicity patterns, their statistical significance and physical meaning*, pp. 471–507, 1999.

Takahiro Omi, Kazuyuki Aihara, et al. Fully neural network based model for general temporal point processes. *Advances in neural information processing systems*, 32, 2019.

Joseph Sill. Monotonic networks. *Advances in neural information processing systems*, 10, 1997.

Wilson Truccolo, Uri T Eden, Matthew R Fellows, John P Donoghue, and Emery N Brown. A point process framework for relating neural spiking activity to spiking history, neural ensemble, and extrinsic covariate effects. *Journal of neurophysiology*, 93(2):1074–1089, 2005.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Matthew Watson, François Chollet, Divyashree Sreepathihalli, Samaneh Saadat, Ramesh Sampath, Gabriel Rasskin, Scott Zhu, Varun Singh, Luke Wood, Zhenyu Tan, Ian Stenbit, Chen Qian, Jonathan Bischof, et al. Kerashub. `https://github.com/keras-team/keras-hub`, 2024.

Simon N Wood. *Generalized additive models: an introduction with R*. chapman and hall/CRC, 2017.

Siqiao Xue, Xiaoming Shi, Zhixuan Chu, Yan Wang, Hongyan Hao, Fan Zhou, Caigao Jiang, Chen Pan, James Y. Zhang, Qingsong Wen, Jun Zhou, and Hongyuan Mei. Easytpp: Towards open benchmarking temporal point processes, 2024. URL `https://arxiv.org/abs/2307.08097`.