

Cumulative Hazard Function Based Efficient Multivariate Temporal Point Process Learning

Bingqing Liu

Academy of Mathematics and Systems Science, CAS
University of Chinese Academy of Science, Beijing, China
liubingqing20@mailsucas.ac.cn

Abstract—Most existing temporal point process models are characterized by conditional intensity function. These models often require numerical approximation methods for likelihood evaluation, which potentially hurts their performance. By directly modelling the integral of the intensity function, i.e., the cumulative hazard function (CHF), the likelihood can be evaluated accurately, making it a promising approach. However, existing CHF-based methods are not well-defined, i.e., the mathematical constraints of CHF are not completely satisfied, leading to untrustworthy results. For multivariate temporal point process, most existing methods model intensity (or density, etc.) functions for each variate, limiting the scalability. In this paper, we explore using neural networks to model a flexible but well-defined CHF and learning the multivariate temporal point process with low parameter complexity. Experimental results on six datasets show that the proposed model achieves the state-of-the-art performance on data fitting and event prediction tasks while having significantly fewer parameters and memory usage than the strong competitors. The source code and data can be obtained from <https://github.com/lbq8942/NPP>.

Index Terms—temporal point process, cumulative hazard function, event sequence modelling

I. INTRODUCTION

Event sequences in continuous time space are ubiquitous around us - hospital visits [1], [2], tweets [3], financial transactions [4], earthquakes [5], human activities [6]–[8], to name a few. Beyond mere timestamps, these sequences usually also carry rich information about the types of events. Inside the event sequence, various types of asynchronous events often exhibit complex hidden dynamics responsible for their generation [9]–[11]. Diving into the dynamics, temporal point process (TPP) [12], [13] has been a popular and principled tool for event sequence modelling, through the lens of which, these sequences can be interpreted as realizations of an underlying stochastic process.

Traditional TPPs often use over-simplified historical dependencies [14]–[16]. For example, Poisson process [17] assumes the future events are independent of the past events. With the development of deep learning, the so-called neural temporal point process (NTPP) models [18]–[22] have been intensively devised for more flexible historical dependencies modeling. NTPP is typically trained by maximizing the likelihood [16], which consists of two terms, the intensity and its integral (cumulative hazard function, CHF). Most existing NTPPs are modeled by conditional intensity functions [16], [23]–[25], by which the intensity term in the likelihood can be

obtained directly. However, due to the high non-linearity of neural networks, the integral term is usually intractable. As a result, approximation methods are required, e.g., Monte Carlo algorithm, trapezoidal rule, Simpson’s rule and etc [16], [24]. Not only is the approximation inefficient, but also the inaccurate likelihood evaluation can potentially harm the training and finally the testing performance. FullyNN [26] pioneered the CHF-based modelling, by which the integral term is directly in hand, and the intensity can be obtained by taking its derivative. Not like integration, the derivation can be exactly calculated. However, FullyNN fails to define a valid CHF, the unfulfilled mathematical constraints make the achieved performance untrustworthy [27]. Moreover, FullyNN defines a bounded CHF, limiting its flexibility.

In the realm of multivariate temporal point process [28], scalability remains a neglected aspect that deserves greater attention. Prior studies usually learn intensity (or density, etc.) functions individually for each variate (event type) [15], [23], [24], which can be prohibitively expensive when dealing with real-world event sequences that encompass numerous event types. In fact, the event observations are so sparse that there’s no need seeing the entire view for all event types.

The contributions of this paper are summarised as follows:

- We present a flexible and well-defined cumulative hazard function network and incorporate it into a parameter-efficient multivariate temporal point process.
- We show that the proposed model performs better in data fitting and predictive tasks for multiple datasets.
- We show that our model has lower parameter complexity and memory footprint, with up to $100 \times$ parameter and memory reduction comparing to state-of-the-art baselines.

II. PRELIMINARIES AND RELATED WORK

Before diving into details, we first introduce some preliminaries to facilitate the presentation of our approach.

A. Temporal Point Process

The dynamics of a TPP can be represented as a counting process $N(t)$ which counts the number of events that happened until time t . A more common definition is by its conditional intensity function:

$$\lambda(t|\mathcal{H}_t) = \lim_{\Delta t \rightarrow 0} \frac{\mathbb{E}(N(t + \Delta t) - N(t)|\mathcal{H}_t)}{\Delta t} \quad (1)$$

which we interpret as: for a short time window $[t, t + \Delta t)$, $\lambda(t)$ represents the rate for the occurrence of a new event conditioned on the past events $\mathcal{H}_t = \{t_i | t_i < t\}$. Given the intensity, the density can be derived as:

$$P_t(t|\mathcal{H}_t) = \lambda(t|\mathcal{H}_t) \exp\left(-\int_{t_{max}}^t \lambda(s|\mathcal{H}_s) ds\right) \quad (2)$$

where t_{max} is the last event time in \mathcal{H}_t . The conditional intensity function has played a central role in point processes and many popular processes differ on how it is parameterized. Classical Poisson process [17] assumes the future events are independent of the history and defines the conditional intensity function as a constant. Hawkes process (self-exciting process) [29] captures the excitation effects from the past events by definition $\lambda(t|\mathcal{H}_t) = \mu + \sum_{t_i} \alpha \exp(-\beta(t - t_i))$. In self-correcting process [30], the past events have inhibition effects on the future events, defined by $\lambda(t|\mathcal{H}_t) = \exp(\mu t - \sum_{t_i} \alpha)$. Traditional point processes make strong assumptions about the latent dynamics, thus having limited expressive power [14].

B. Neural Point Process

An important aspect for temporal point process is to model the historical dependencies [14], [24], [31]. To be more flexible, neural networks are introduced to encode the events history, such as recurrent neural network (RNN) and Transformer [32], [33]. The neural encoder maps a historical event sequence to a low-dimensional vector, based on which conditional intensity function (or other functions like the density function [27]) is defined. RMTTP [14] uses RNN to encode the past events as vector h and defines $\lambda(t|\mathcal{H}_t) = \exp(v^T h + \alpha(t - t_{max}) + b)$, where t_{max} is the last event time in \mathcal{H}_t . Some other works have a similar definition [24], [25], [34]. These models are not flexible because they actually specified a monotonic intensity function. To enhance the flexibility, NHP [16] proposes a novel RNN architecture - Continuous-time LSTM to model a time-varying hidden vector $h(\cdot)$ and defines the intensity function as $\lambda(t|\mathcal{H}_t) = \sigma(v^T h(t - t_{max}))$, where σ is an activation function to ensure the intensity is non-negative. UNIPoint [23] fits a complex intensity function by adding up a bunch of parametric simple functions, specifically $\lambda(t|\mathcal{H}_t) = \sigma(\sum_{c=1}^C \alpha_c \exp(\beta_c(t - t_{max})))$, where α_c, β_c are parameters learned from history vector h . Theoretically elegant, UNIPoint is proven to have the universal approximation property. While these models offer considerable flexibility, they come with a tradeoff for closed-form likelihood evaluation. As a result, approximation methods such as Monte Carlo are required, leading to inaccuracy and inefficiency. To overcome this disadvantage, FullyNN [26] proposes to directly model the integral of the intensity function (cumulative hazard function). Technically, FullyNN uses a neural network with positive parameters [35]–[37] to satisfy the monotonic constraint of cumulative hazard function. In FullyNN, the likelihood can be computed accurately and the flexibility is guaranteed by the neural network. Beyond the monotonicity, there actually exist some other mathematical constraints for cumulative hazard function, which however, are not fulfilled by FullyNN. This

makes the performance obtained from FullyNN implausible. LNM [27] proposes to model the probability density function by mixture log-normal distribution, which is able to approximate any complex distribution. The major drawback of LNM is its overfitting problem, which requires variance clipping technique to ensure numerically stable training and inference.

Multi-type event sequences are ubiquitous across applications, which can be correspondingly characterized by multivariate temporal point process [28]. Most existing methods define intensity functions [16], [23] $\lambda_k(t|\theta_k, \mathcal{H}_t)$ (or cumulative hazard functions [26], probability density functions [15], [27], [38], etc.) for each event type k , with different types of intensity functions (or other functions) having the same parametric form but different parameters θ_k . By increasing the number of parameters in θ_k , the model can indeed exhibit greater flexibility. However, this tends to be prohibitively expensive in scenarios involving numerous event types. RMTTP [14] scales well but has inferior performance for it only defines a simple intensity function and doesn't consider the inter-dependence between time and event type.

III. METHOD

A. Multivariate Temporal Point Process

Given an observed multi-type event sequence $S = \{(m_1, t_1), \dots, (m_N, t_N)\}$, where $m \in \{1, \dots, M\}$, $t \in \mathbb{R}^+$ denote the event type and time, event types are categorical and event times are strictly increasing in continuous space, multivariate temporal point process (MTPP) is essentially modelling the conditional joint distribution $P(m, t|\mathcal{H}_t)$, where history $\mathcal{H}_t = \{(m_i, t_i) | t_i < t\}$. Naturally, MTPP model can be learned by maximizing the log-likelihood below:

$$\mathcal{L}_l = \sum_{i=1}^N \log P(m_i, t_i|\mathcal{H}_{t_i}) \quad (3)$$

Given condition \mathcal{H}_{t_i} , we can equivalently model the inter-event time $\tau_i = t_i - t_{i-1}$. For simplicity, we will omit the writing of history condition \mathcal{H}_{t_i} from now on for we are always modelling conditional distributions throughout this paper. A widely adopted joint distribution modelling strategy is to characterize the intensity functions, i.e.,

$$\begin{aligned} P(m_i, \tau_i) &= P_t(\tau_i) P_m(m_i|\tau_i) \\ &= [\lambda(\tau_i) \exp(-\int_0^{\tau_i} \lambda(s) ds)] \left[\frac{\lambda_{m_i}(\tau_i)}{\lambda(\tau_i)} \right] \\ \lambda(\tau_i) &= \lambda_1(\tau_i) + \dots + \lambda_M(\tau_i) \end{aligned} \quad (4)$$

where $P_t(\cdot)$ is probability density function defined on \mathbb{R}^+ , $P_m(\cdot)$ is a M -categorical distribution and $\lambda(\cdot)$ is intensity function. In this framework, all intensity functions $\{\lambda_m(\cdot)\}_{m=1}^M$ are learning objectives, which has a high model complexity especially when M is particularly large. Moreover, the log-likelihood contains an integral with respect to the intensity function, which is always intractable and requires approximation methods such as Monte Carlo in practice, leading to inaccurate and inefficient model learning.

To overcome these shortcomings, we propose to model the joint distribution by only learning the total intensity function and the time-dependent type distribution, i.e.,

$$P(m_i, \tau_i) = \lambda(\tau_i) \exp\left(-\int_0^{\tau_i} \lambda(s) ds\right) P_m(m_i|\tau_i) \quad (5)$$

As a comparison, the aforementioned framework (4) needs learning M continuous variables $\{\lambda_m(\cdot)\}_{m=1}^M$ while we only have one continuous variable $\lambda(\cdot)$ and one discrete variable $P_m(\cdot|\tau_i)$, which greatly reduces the learning complexity. To avoid the integral computation in (5), we borrow ideas from FullyNN [26]: we directly model the integral function $\int_0^\tau \lambda(s) ds$, named cumulative hazard function (CHF). Accordingly, the intensity function can be derived through gradient calculation, i.e., $\partial \int_0^\tau \lambda(s) ds / \partial \tau$, which can be automatically and exactly calculated by existing deep learning library (e.g., Pytorch). As a result, the proposed method can evaluate the likelihood accurately and efficiently.

B. Model Architecture

To model the joint distribution $P(m_{i+1}, \tau_{i+1})$, the first step is to encode the past events $\mathcal{H}_{t_{i+1}}$ into a low-dimensional dense semantic vector h_i . Commonly used event sequence encoder includes recurrent neural network (RNN [39]) and Transformer [32]. In this paper, we choose RNN as our encoder for it has competitive performance but fewer parameters. RNN recursively updates the history vector h by digesting new coming events, which can be formulated as follows,

$$h_i = \text{update}(h_{i-1}, e_i) \quad (6)$$

where $e_i \in \mathbb{R}^{d_m+1}$ is the embedding of the i -th event (m_i, t_i) , which is specified as the concatenation of the trainable type embedding $\mathbf{m}_i \in \mathbb{R}^{d_m}$ and the inter-event time τ_i , telling the encoder which type of event happens and how long has it been since the last event. The operation *update* in (6) in vanilla RNN is formulated as follows,

$$h_i = \tanh(W_h(h_{i-1} || e_i) + b_h) \quad (7)$$

where $||$ denotes the concatenation of two vectors, \tanh is the activation function for non-linearity, weight matrix $W_h \in \mathbb{R}^{d \times (d_m+1)}$ (d is the hidden dimension for RNN) and bias vector $b_h \in \mathbb{R}^d$ are trainable parameters. More complicated *update* for (6), e.g., GRU [39] and LSTM [40], can also be used for potentially better performance.

The second step is to model the cumulative hazard function $\int_0^\tau \lambda(s) ds$ and the time-dependent type distribution $P_m(m|\tau)$ in (5). The type distribution is categorical, conditioned on past events \mathcal{H}_{t_i} and the occurrence time. Thus, we model the type distribution as a M -dimensional vector that has history- and time-varying elements. Specifically, we use a two-layer neural network, i.e.,

$$\begin{aligned} P_m(m|\tau) &= \text{softmax}(W_m^2 y_m^2 + b_m^2) \\ y_m^2 &= \text{relu}(W_m^1(h_i || \tau) + b_m^1) \end{aligned} \quad (8)$$

where weight matrices and biases $W_m^1 \in \mathbb{R}^{d \times (d+1)}$, $W_m^2 \in \mathbb{R}^{M \times d}$, $b_m^1 \in \mathbb{R}^d$, $b_m^2 \in \mathbb{R}^M$ are all trainable parameters, *relu*

is the rectified linear unit, defined by $\max(0, \cdot)$, *softmax* is a normalizing function that makes an arbitrary M -dimensional vector a categorical probability, with each element's exponent divided by the exponential sum.

The cumulative hazard function $\phi(\tau) = \int_0^\tau \lambda(s) ds$ is monotonic with respect to the inter-event time τ since the intensity function is non-negative. FullyNN [26] simulates the CHF by a neural network where the neural weights are all positive and the activation functions are all monotonic. However, neural network defined that way doesn't meet the requisite $\phi(0) = 0$, making it a flawed design. To repair it, we try not posing any extra restrictions or doing any modifications on the network architecture so as to not harm the expressiveness. For this sake, we reproduce a well-defined CHF network by dynamically subtracting the initial bias $\phi(0)$. We say the bias is dynamic because it's history-dependent rather than a fixed something (e.g., network architecture and its parameters). The whole cumulative hazard function network (three-layer for example) is then defined as,

$$\begin{aligned} \phi(\tau) &= f(\tau) - f(0) \\ f(\tau) &= v_t^T z_t^3 + b_t^3 \\ z_t^3 &= \sigma_t(W_t^2 z_t^2 + b_t^2) \\ z_t^2 &= \sigma_t(W_t^1(h_i || \tau) + b_t^1) \end{aligned} \quad (9)$$

where neural weights $W_t^2 \in \mathbb{R}^{d \times d}$, $v_t \in \mathbb{R}^d$ in the second and third layer are constrained to be positive. For the weight matrix $W_t^1 \in \mathbb{R}^{d \times (d+1)}$ in the first layer, the last column vector that acts on t is also required to be positive. For the remaining parameters in W_t^1 and biases $b_t^1 \in \mathbb{R}^d$, $b_t^2 \in \mathbb{R}^d$, b_t^3 , we don't pose the positive constraint. In addition to the constraints on parameters, activation function σ_t is supposed to be monotonic to ensure the monotonicity of the whole network. We use *tanh* function for this purpose. We have also tried some other commonly used monotonic activation functions such as *relu* and *softplus* but empirically see inferior performance, which we will show in the experiment. So far, we have designed a well-defined CHF network by (9).

The activation function *tanh* is highly non-linear and can activate both positive and negative numbers. However, *tanh* is a bounded function, with the gradient vanished when saturated. The gradient vanishing problem makes the intensity function converges to zero after a period of time, which may conflict with reality. In fact, many events happen not only because of the exogenous factors (excitation or inhibition from other events) but also by endogenous ones (occur spontaneously without external intervention), which is called base intensity and usually modelled as a constant in literature [14], [24], [29]. To capture this pattern, we enhance the flexibility of our CHF network by adding a history-dependent linear residual. Specifically, we modify $\phi(\tau)$ in (9) as follows,

$$\begin{aligned} \phi(\tau) &= f(\tau) - f(0) + g(\tau) \\ g(\tau) &= \gamma \tau, \gamma = \sigma_r(v_\gamma^T h_i + b_\gamma) \end{aligned} \quad (10)$$

where $v_\gamma \in \mathbb{R}^d$, b_γ are parameters to be trained and σ_r is the *relu* activation function, which is non-negative and can ensure

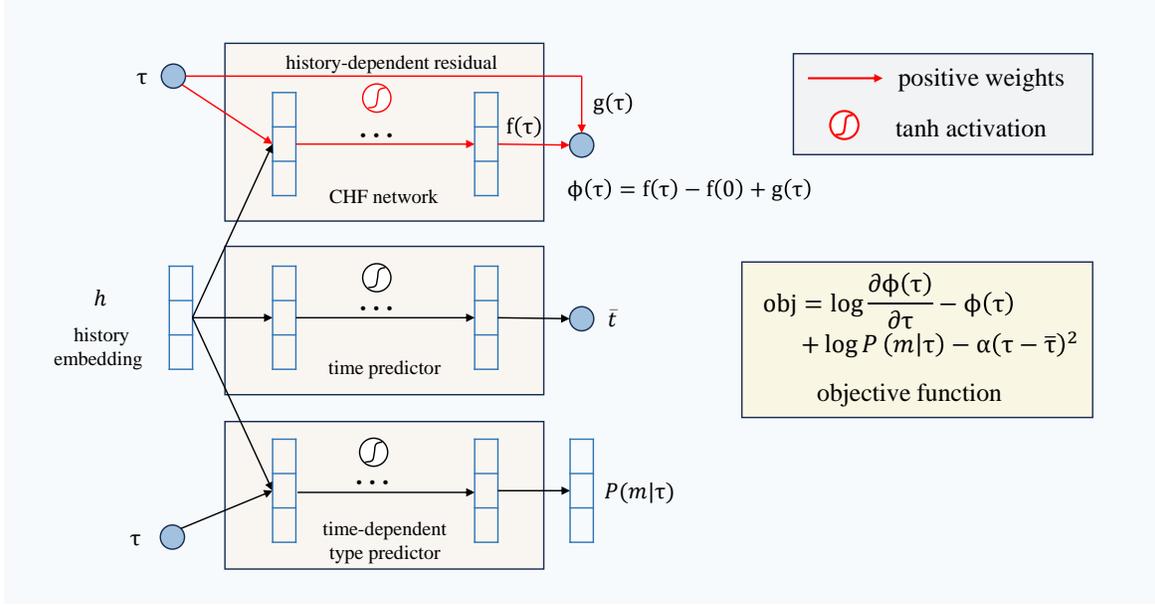


Fig. 1. The proposed cumulative hazard function (CHF) based multivariate temporal point process model, including three modules: the CHF network, the time predictor and the time-dependent type predictor. The CHF network is to reproduce a well-defined CHF so as to capture the temporal dynamics, the time predictor is for time prediction, and the time-dependent type predictor is to capture the dynamics of event types and also used for type prediction.

the monotonicity of the CHF network $\phi(\tau)$. With the added residual, the intensity function $\partial\phi(\tau)/\partial\tau$ now converges to the base intensity γ . Notably, the residual can adapt itself to the history, gaining more flexibility. If γ is learned to be 0, then the CHF network defined in (10) degrades to that in (9).

When doing prediction on the $(i+1)$ -th event, the proposed model should first predict the event time and then predict the event type according to (5). The event time is predicted via the CHF network $\phi(\tau)$ and the event type is predicted through the time-dependent type predictor $P_m(m|\tau)$. Specifically, if we use the expectation $\int_0^\infty sP_t(s)ds$ as the predicted time $\hat{\tau}$, then according to (2) we have the following predicting procedure,

$$\begin{aligned} \hat{\tau} &= \int_0^\infty s \frac{\partial\phi(s)}{\partial s} \exp(-\phi(s)) ds \\ \hat{m} &= \underset{\circ}{\operatorname{argmax}} P_m(\circ|\hat{\tau}) \end{aligned} \quad (11)$$

The inconvenience for the time prediction is that the expectation $\hat{\tau}$ often lacks an analytical solution. Consequently, numerical estimation methods become necessary, which not only have the potential to introduce inaccuracies in the prediction results but can also be computationally intensive. To avoid the usage of approximation methods and make fast time prediction, we additionally add a time predictor, realized by a two-layer neural network, i.e.,

$$\bar{\tau} = v_p^T (W_p^1 h_i + b_p^1) + b_p^2 \quad (12)$$

where $W_p^1 \in \mathbb{R}^{d \times d}$, $b_p^1 \in \mathbb{R}^d$, $v_p^T \in \mathbb{R}^d$ and b_p^2 are trainable parameters. Time predictor takes history vector h_i as input and straightforwardly outputs the predicted time for the $(i+1)$ -th event. The whole model architecture is illustrated in Fig. 1.

C. Training Loss

Training a temporal point process model is to maximize its log-likelihood on observed event sequences. Follow the notation in Section III-A, given the observed sequence S , the log-likelihood can be computed as follows,

$$\begin{aligned} \mathcal{L}_t &= \frac{1}{N} \sum_{i=1}^N [\log P^*(m_i, \tau_i) = \log P_t^*(\tau_i) + \log P_m^*(m_i|\tau_i)] \\ &= \log \left. \frac{\partial\phi^*(\tau)}{\partial\tau} \right|_{\tau=\tau_i} - \phi^*(\tau_i) + \log P_m^*(m_i|\tau_i) \end{aligned} \quad (13)$$

where superscript $*$ reminds us of the history condition H_{t_i} . The performance on time prediction by the time predictor (TP) in (12) is evaluated by the mean square error, i.e.,

$$\mathcal{L}_t = \frac{1}{N} \sum_{i=1}^N (\tau_i - \text{TP}(h_{i-1}))^2 \quad (14)$$

We learn the whole model under the multi-task learning framework. Therefore, we construct the final loss as $\mathcal{L} = -\mathcal{L}_l + \alpha\mathcal{L}_t$. The first term is called negative log-likelihood (NLL) and α is a hyperparameter that controls the loss from time prediction.

IV. EXPERIMENT

A. Datasets

We evaluate the proposed model over two synthetic datasets Hawkes1, Hawkes2, and four real-world datasets Retweets [24], SOflow [15], MIMIC [15] and Social [8]. Hawkes1, Hawkes2 are simulated by the Hawkes process, where past events will excite future events and the influences decay exponentially over time. Hawkes1 and Hawkes2 have different

TABLE I
DATASET STATISTICS.

Dataset	#Types	#Seqs	Avg. Length	Interval
Hawkes1	9	9494	7.4	0.51(0.55)
Hawkes2	9	3960	10.4	0.28(0.50)
Retweet	3	20000	108.8	3.98(2.60)
SOflow	22	6633	72.2	12.72(1.83)
Mimic	75	715	3.7	0.41(0.38)
Social	1641	1337	99.0	1.18(1.66)

parameters that control the influences.¹ The four real-world datasets cover different application domains, collecting events from social platform, question answering website, healthcare records and offline social activity, respectively. Each dataset has a number of event sequences and sequences can have different length. These statistics together with the number of event types, mean inter-event time, and its standard deviation (in bracket) are reported in Table I.

B. Baselines

To evaluate the effectiveness of our model, we compare it against the state-of-the-art models. We categorize the selected baselines into two classes, intensity-based and -free models. We choose RMTTP [14], NHP [16], SAHP [25], THP [24] and UNIPoint [23] for intensity-based models, where SAHP and THP utilize Transformer as the history encoder and the others use the recurrent neural network. There are fewer studies concerning the non-intensity based temporal point process, we choose FullyNN and JTPP [15] (extended from LNM [27] for multivariate TPP) as baselines, they both achieve the best performance in their paper.

C. Implementation Details

We use Adam algorithm to optimize the parameters, with the learning rate set to 0.001 and the hyperparameter α in the loss function set to 0.01 for all six datasets. Mini-batch algorithm is adopted for training, with 64 event sequences per batch. Sequences in one batch are padded to the maximum length. To ensure the τ -related parameters in cumulative hazard function network to be positive, we technically take their absolute values each time the parameters are updated. Each dataset is split into training, validation and testing data, with the proportion being 0.6, 0.2 and 0.2, respectively. The embedding dimension for event types (d_m) and the hidden dimension for history vector (d) is set to 64 and 128 for dataset Social, 32 and 64 for other datasets. We adopt the early stopping strategy, i.e., stop training when performance on the validation data has not been improved for 10 epochs. The baseline models also apply these setups if exist. The training is conducted on GPU NVIDIA 3090 with 24GB for all models.

D. Data Fitting

Likelihood indicates how a temporal point process model fits the observed sequences. Table II summarizes the negative log-likelihoods (per sequence) on testing data for all

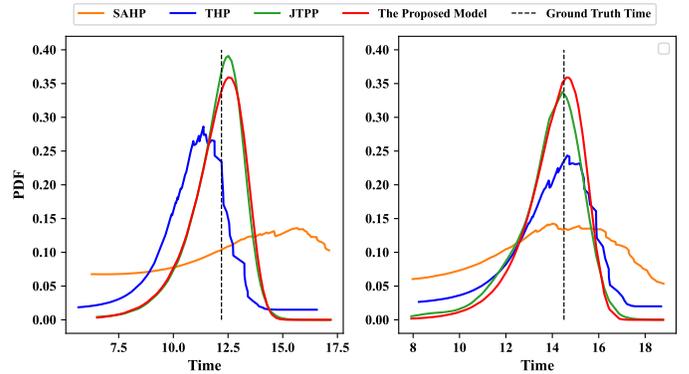


Fig. 2. Case study: two learned probability density functions on dataset SOflow. The proposed model and JTPP fit the ground truth time more exactly.

the models. Lower negative log-likelihood indicates better performance. FullyNN is denoted with a strikeout, reminding us that this model does not define a valid CHF. As a result, FullyNN has biased results and should not be compared for fairness. Some baselines encounter out of memory on our provided 24GB GPU over dataset Social (a dataset with 1641 event types). We denote this phenomenon as OOM. Table II shows that the proposed model achieves the best performance for 5 of 6 datasets, and is the second best on dataset Retweet, which validates our effectiveness. JTPP obtains the second best performance, but can not be applied to datasets with thousands of event types. Both the proposed model and JTPP are non-intensity based, where the log-likelihood can be calculated exactly for training. As a contrast, state-of-the-art intensity-based models such as UNIPoint and NHP require numerical estimation, which hurts their performance though UNIPoint has been theoretically proven to have the universal approximation property. Other baselines including RMTTP, SAHP and THP specify the intensity function to be monotonically decreasing or increasing, thus the flexibility is limited. In our model, the cumulative hazard function is characterized by a neural network, which in principle can be a much more flexible function. A case study is conducted on dataset SOflow to see the learned probability density function. TPP models generate conditional density function at each event position, two of them are shown in Fig. 2. We see that the proposed model and JTPP fit the ground truth time more exactly: the learned density function reaches its peak near the ground truth time and has a smaller entropy.

E. Event prediction

TPP models encode the past events and generate the conditional distribution for the next event, which can be leveraged for event prediction, including the type prediction and the occurrence time prediction. The predicting procedure for the proposed model has been described in (11) and (12). For event type prediction, we use the weighted F1 score to evaluate the performance, and for event time prediction we use the mean absolute error. Higher F1 score and lower time

¹The details can be found at <https://github.com/lbq8942/NPP>

TABLE II

NEGATIVE LOG-LIKELIHOOD (PER SEQUENCE) COMPARISON. THE LOWER THE BETTER. THE RESULTS ARE AVERAGED BY 10 RUNS AND THE STANDARD DEVIATION IS REPORTED IN BRACKET. THE BEST AND SECOND BEST MODEL ON EACH DATASET HAVE BEEN HIGHLIGHTED IN BOLD AND UNDERLINE RESPECTIVELY. OOM MEANS OUT OF MEMORY ON 24GB GPU.

Model	Hawkes1	Hawkes2	Retweet	SOflow	MIMIC	Social
RMTTP	11.3(0.0)	10.1(0.1)	262.8(2.0)	246.0(3.3)	7.3(1.4)	<u>354.6(26.9)</u>
NHP	<u>9.7(0.0)</u>	9.4(0.0)	253.7(2.3)	238.2(2.9)	7.1(1.3)	OOM
SAHP	10.2(0.0)	9.6(0.1)	272.8(7.5)	235.4(3.2)	6.9(1.6)	478.7(7.5)
THP	10.4(0.0)	9.8(0.1)	264.0(1.9)	237.4(1.4)	6.8(1.7)	488.9(9.8)
UNIPoint	9.9(0.0)	9.4(0.0)	254.9(1.9)	232.3(2.9)	5.7(1.2)	OOM
JTPP	10.0(0.0)	9.3(0.1)	235.4(2.6)	225.3(2.7)	5.8(1.2)	OOM
FullyNN	<u>9.9(0.1)</u>	<u>9.7(0.0)</u>	<u>238.0(3.2)</u>	<u>233.8(6.2)</u>	6.7(2.1)	OOM
<i>ours</i>	9.6(0.1)	9.0(0.0)	<u>240.4(4.8)</u>	223.9(2.5)	5.6(1.2)	256.5(8.6)

TABLE III

EVENT TYPE PREDICTION ACCURACY (WEIGHTED F1 SCORE) COMPARISON. THE HIGHER THE BETTER.

Model	Hawkes1	Hawkes2	Retweet	SOflow	MIMIC	Social
RMTTP	37.2(0.6)	22.6(0.6)	59.1(0.0)	31.2(0.2)	63.2(4.5)	43.1(4.6)
NHP	38.6(0.3)	24.7(0.6)	59.6(0.1)	31.0(0.8)	63.4(4.8)	OOM
SAHP	36.8(0.3)	22.9(0.2)	58.4(1.2)	31.5(1.0)	65.3(5.8)	21.0(0.2)
THP	38.5(0.5)	23.8(0.4)	59.8(0.2)	31.6(0.8)	65.5(5.6)	29.9(2.0)
UNIPoint	37.8(1.1)	24.0(0.8)	59.7(0.0)	30.7(0.4)	66.0(5.2)	OOM
JTPP	39.0(0.3)	<u>24.5(0.9)</u>	60.0(0.1)	31.8(0.2)	65.4(5.5)	OOM
<i>ours</i>	<u>38.7(0.5)</u>	24.7(0.6)	60.2(0.1)	31.6(0.2)	65.7(4.6)	45.5(0.5)

TABLE IV

EVENT TIME PREDICTION ERROR (MEAN ABSOLUTE ERROR) COMPARISON. THE LOWER THE BETTER.

Model	Hawkes1	Hawkes2	Retweet	SOflow	MIMIC	Social
RMTTP	0.3755(0.0028)	0.2238(0.0007)	1.9268(0.0067)	4.1043(0.1192)	0.4475(0.0252)	<u>0.9848(0.0481)</u>
NHP	0.3704(0.0025)	0.2160(0.0028)	1.8814(0.0056)	5.0963(0.0492)	0.4063(0.0182)	OOM
SAHP	0.3748(0.0045)	0.2154(0.0037)	1.8160(0.0495)	2.9639(0.4214)	0.3335(0.0130)	1.2193(0.0037)
THP	<u>0.3681(0.0039)</u>	0.2116(0.0031)	1.8902(0.0082)	5.1592(0.0315)	0.3546(0.0124)	1.0321(0.0288)
UNIPoint	0.3695(0.0030)	0.2151(0.0021)	1.8384(0.0030)	4.5776(0.0839)	0.2462(0.0046)	OOM
JTPP	0.3786(0.0268)	0.1961(0.0252)	<u>1.0245(0.0021)</u>	1.2557(0.0178)	0.2755(0.0081)	OOM
<i>ours</i>	0.3617(0.0030)	0.1587(0.0028)	1.0158(0.0022)	1.1722(0.0018)	<u>0.2511(0.0053)</u>	0.7575(0.0251)

error indicate better performance. The results for these two predictions are reported in Table III and IV, respectively. Table III shows that the proposed model outperforms the baselines, though not significantly against JTPP (excluding dataset Social). However, JTPP scales not well to datasets with hundreds and thousands of event types, limiting its application. In Table III, an interesting phenomenon is that the proposed model seems to have more stable performance, while most baselines perform not well on at least one dataset, e.g., Social. This is because for most baselines, the event type prediction is time-dependent, which means inaccurate time prediction will badly affect the type prediction. The time prediction results in Table IV show that we have the least prediction errors in most datasets while most baselines fall short on the time prediction on dataset SOflow, which as a result causes the error propagation to the type prediction. RMTTP and JTPP both predict the event type by a multi-layer perceptron type predictor (not time-dependent). Hence the time prediction results will not affect their type prediction. We insist the time-dependent type modelling for it's more expressive and could potentially witness more applications: consider the situation where the ground truth time is given or we would like to analyze how type distribution evolves

over time. In contrast, the time-independent type predictor can not utilize the timestamps information and thus will be fairly disadvantageous in these cases. For time prediction, we see the proposed model has a significant error reduction compared with baselines on most datasets. Considering both type and time prediction results in Table III and IV, they highlight the superiority of our proposed model.

F. Model Complexity

For our mainly focused parameter complexity, the proposed method only models a cumulative hazard function and a time-dependent type predictor for multivariate temporal point process. As a comparison, most baseline methods learn intensity (or density, etc.) functions tailored to each individual event type. These approaches typically involve a substantial number of parameters, particularly in instances where an intricate intensity (or density, etc.) function is designed for flexibility, as exemplified by UNIPoint and JTPP. Fig. 3 compares the trade-off between the fitting performance and required parameters for all baselines and the proposed model on four datasets. The standardized NLL is the negative log-likelihood in Table II divided by the average sequence length (see Table I). NHP, UNIPoint and JTPP can not run on dataset Social due to out of memory, hence we only report their required parameters.

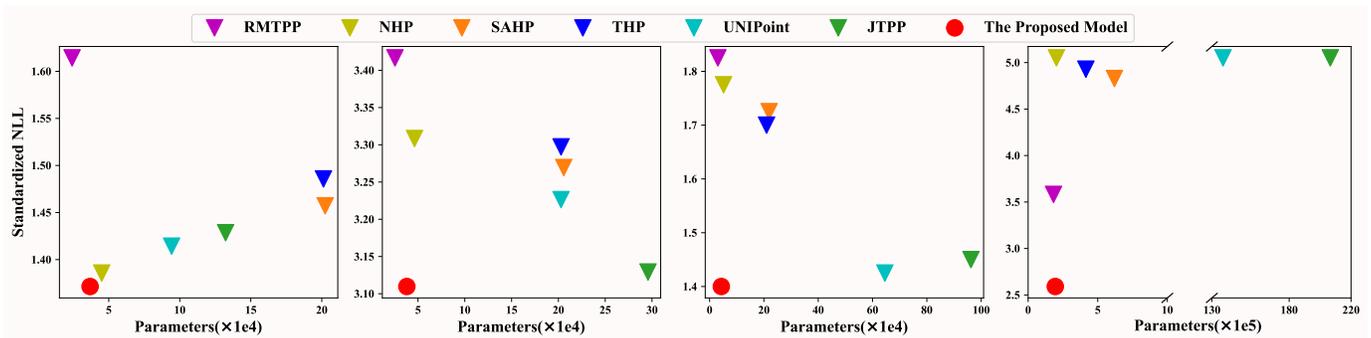


Fig. 3. The fitting performance and required parameters for baselines and the proposed model on datasets Hawkes1, SOflow, MIMIC and Social. In each plot, models locate in the bottom and left are better.

TABLE V
PEAK GPU MEMORY USAGE (GB).

Model	Hawkes1	Hawkes2	Retweet	SOflow	MIMIC	Social
RMTTP	0.01	0.02	0.05	0.14	0.01	0.23
NHP	0.26	0.39	0.55	5.80	0.80	OOM
SAHP	0.14	0.25	1.02	6.81	0.17	11.63
THP	0.15	0.25	1.02	6.79	0.18	12.78
UNIPoint	0.55	0.85	0.75	14.61	2.24	OOM
JTPP	0.15	0.23	0.22	3.82	0.59	OOM
ours	0.03	0.04	0.11	0.32	0.02	0.28

TABLE VI

THE INFLUENCE OF THE HYPERPARAMETER α IN THE LOSS FUNCTION, EVALUATED ON DATASET RETWEET.

Task	$\alpha=0.001$	$\alpha=0.01$	$\alpha=0.1$	$\alpha=1$
data fitting	240.1	240.4	243.5	271.5
type prediction	60.3	60.2	60.2	60.1
time prediction	1.0146	1.0135	1.0115	1.0123

Fig. 3 shows that the proposed model not only achieves the best fitting performance but also has almost the least parameters, making it more desirable for multi-type event sequence modelling. In addition to the required parameters, we also report the peak GPU memory usage in Table V. Note that the memory usage is not only affected by the required parameters but also can be affected by some other factors like batch size, sequence length, model architecture, etc. Except for the model architecture, all other factors are controlled to be the same, as introduced in Section IV-C. Table V again demonstrates the superiority of our proposed model, which enables up to $100\times$ reduction in memory usage comparing to the state-of-the-art baselines.

G. Ablation Studies

In this part, we investigate the impact of hyperparameter α in the loss function and the impact of the activation function σ_t in the cumulative hazard function network. Both of them are conducted on dataset Retweet. The hyperparameter α controls the loss from the time prediction and is set to 0.01 by default. To see how different α affect the model performance, we set it from candidate set $\{0.001, 0.01, 0.1, 1\}$. The results are shown in Table VI, from which we can see that the performance on

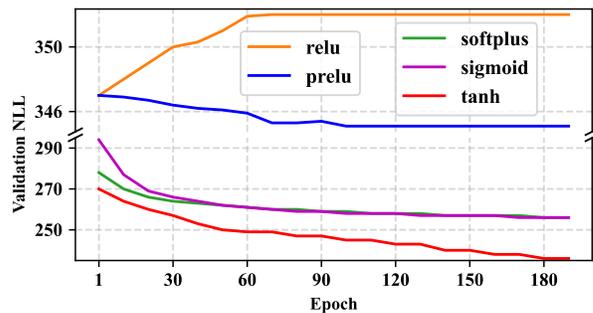


Fig. 4. Training curves of the proposed model using different activation functions for the cumulative hazard function network, fitted on dataset Retweet.

data fitting gets worse when loss from the time prediction more and more dominates the entire loss. But for event prediction, the performance is surprisingly nearly not affected, showing the robustness.

We show how activation function σ_t in the CHF network will dramatically influence the model performance and commonly used activation functions such as *relu* will lead to inferior performance. Besides *relu*, we also try some other functions that can satisfy the monotonic constraint such as *prelu*, *softplus* and *sigmoid*, where *prelu* is defined as $\max(\eta_1 x, x)$ and *softplus* is defined as $(1/\eta_2) \log(1 + \exp(\eta_2 x))$ (both η_1 and η_2 are trainable). The investigation results are shown in Fig. 4, where we illustrate the learning curves when using different activation functions. The plots show that *tanh* is the best of those tried activation functions and *relu*, *prelu* can not fit the data well. We analyze the reason as that *tanh* can provide more non-linearity and its negative activation can reserve more capacity, both of which enhance the fitting ability.

V. CONCLUSION

In this study, we present a cumulative hazard function based multivariate neural point process model. By modeling CHF, the proposed model enjoys accurate and efficient likelihood evaluation. We design a neural network architecture by which the cumulative hazard function can be well-defined

and flexibly modelled. For multivariate modeling, we adopt a parameter-efficient strategy, making our model more scalable. Extensive experiments are conducted to demonstrate the effectiveness of the proposed model: gains better performance while many fewer parameters and memory usage are required.

REFERENCES

- [1] G. Mavroudeas, N. Neehal, X. Shou, M. Magdon-Ismael, J. N. Kuruzovich, and K. P. Bennett, "Predictive modeling for complex care management," *2021 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pp. 1443–1448, 2021. [Online]. Available: <https://api.semanticscholar.org/CorpusID:245934223>
- [2] E. Choi, N. Du, R. Chen, L. Song, and J. Sun, "Constructing disease network and temporal progression model via context-sensitive Hawkes process," *2015 IEEE International Conference on Data Mining*, pp. 721–726, 2015. [Online]. Available: <https://api.semanticscholar.org/CorpusID:1370873>
- [3] J. ge Yao, F. Fan, W. X. Zhao, X. Wan, E. Y. Chang, and J. Xiao, "Tweet timeline generation with determinantal point processes," in *AAAI Conference on Artificial Intelligence*, 2016. [Online]. Available: <https://api.semanticscholar.org/CorpusID:37498580>
- [4] E. Bacry, I. Mastromatteo, and J.-F. Muzy, "Hawkes processes in finance," *Market Microstructure and Liquidity*, vol. 1, no. 01, p. 150005, 2015.
- [5] M. N. M. van Lieshout and A. Stein, "Earthquake modelling at the country level using aggregated spatio-temporal point processes," *Mathematical Geosciences*, vol. 44, pp. 309–326, 2012.
- [6] K. Zhou, H. Zha, and L. Song, "Learning social infectivity in sparse low-rank networks using multi-dimensional Hawkes processes," in *International Conference on Artificial Intelligence and Statistics*, 2013. [Online]. Available: <https://api.semanticscholar.org/CorpusID:8326502>
- [7] M. Farajtabar, Y. Wang, M. Gomez Rodriguez, S. Li, H. Zha, and L. Song, "Coevolve: A joint point process model for information diffusion and network co-evolution," *Advances in Neural Information Processing Systems*, vol. 28, 2015.
- [8] B. Liu and X. Huang, "Link-aware link prediction over temporal graph by pattern recognition," in *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*, 2023. [Online]. Available: <https://api.semanticscholar.org/CorpusID:260170307>
- [9] Q. Zhang, A. Lipani, and E. Yilmaz, "Learning neural point processes with latent graphs," *Proceedings of the Web Conference 2021*, 2021.
- [10] F. Ding, J. Yan, and H. Wang, "C-ntpp: Learning cluster-aware neural temporal point process," in *AAAI Conference on Artificial Intelligence*, 2023. [Online]. Available: <https://api.semanticscholar.org/CorpusID:259739114>
- [11] Y. Zhang and J. Yan, "Neural relation inference for multi-dimensional temporal point processes via message passing graph," in *International Joint Conference on Artificial Intelligence*, 2021. [Online]. Available: <https://api.semanticscholar.org/CorpusID:237100690>
- [12] D. J. Daley, D. Vere-Jones *et al.*, *An introduction to the theory of point processes: volume I: elementary theory and methods*. Springer, 2003.
- [13] D. J. Daley and D. Vere-Jones, *An introduction to the theory of point processes: volume II: general theory and structure*. Springer Science & Business Media, 2007.
- [14] N. Du, H. Dai, R. Trivedi, U. Upadhyay, M. Gomez-Rodriguez, and L. Song, "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*, 2016, pp. 1555–1564.
- [15] G. Waghmare, A. Debnath, S. Asthana, and A. Malhotra, "Modeling inter-dependence between time and mark in multivariate temporal point processes," in *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, 2022, pp. 1986–1995.
- [16] H. Mei and J. M. Eisner, "The neural Hawkes process: A neurally self-modulating multivariate point process," *Advances in Neural Information Processing Systems*, vol. 30, 2017.
- [17] J. F. C. Kingman, *Poisson processes*. Clarendon Press, 1992, vol. 3.
- [18] O. Shchur, A. C. Türkmen, T. Januschowski, and S. Günnemann, "Neural temporal point processes: A review," *ArXiv*, vol. abs/2104.03528, 2021.
- [19] S. Xue, X. Shi, Z. Chu, Y. Wang, H. Hao, F. Zhou, C. JIANG, C. Pan, J. Y. Zhang, Q. Wen *et al.*, "Easytp: Towards open benchmarking temporal point processes," in *The Twelfth International Conference on Learning Representations*, 2023.
- [20] H. Lin, C. Tan, L. Wu, Z. Gao, S. Li *et al.*, "An empirical study: Extensive deep temporal point process," *ArXiv*, vol. abs/2110.09823, 2021.
- [21] H. Lin, C. Tan, L. Wu, Z. Gao, and S. Z. Li, "Extensive deep temporal point process," *ArXiv*, vol. abs/2110.09823, 2021.
- [22] J. Yan, "Recent advance in temporal point process: from machine learning perspective," *SJTU Technical Report*, 2019.
- [23] A. Soen, A. Mathews, D. Gritti-Cheng, and L. Xie, "Unipoint: Universally approximating point processes intensities," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 11, 2021, pp. 9685–9694.
- [24] S. Zuo, H. Jiang, Z. Li, T. Zhao, and H. Zha, "Transformer Hawkes process," in *International Conference on Machine Learning*. PMLR, 2020, pp. 11 692–11 702.
- [25] Q. Zhang, A. Lipani, O. Kirnap, and E. Yilmaz, "Self-attentive Hawkes process," in *International Conference on Machine Learning*. PMLR, 2020, pp. 11 183–11 193.
- [26] T. Omi, K. Aihara *et al.*, "Fully neural network based model for general temporal point processes," *Advances in Neural Information Processing Systems*, vol. 32, 2019.
- [27] O. Shchur, M. Biloš, and S. Günnemann, "Intensity-free learning of temporal point processes," in *International Conference on Learning Representations*, 2020.
- [28] D. R. Cox and P. Lewis, "Multivariate point processes," *Point Processes*, 2018. [Online]. Available: <https://api.semanticscholar.org/CorpusID:118739078>
- [29] A. G. Hawkes, "Spectra of some self-exciting and mutually exciting point processes," *Biometrika*, vol. 58, no. 1, pp. 83–90, 1971.
- [30] V. Isham and M. Westcott, "A self-correcting point process," *Stochastic Processes and their Applications*, vol. 8, no. 3, pp. 335–347, 1979.
- [31] S. Dash, X. She, and S. Mukhopadhyay, "Learning point processes using recurrent graph network," in *2022 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2022, pp. 1–8.
- [32] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in Neural Information Processing Systems*, vol. 30, 2017.
- [33] Y. Gu, "Attentive neural point processes for event forecasting," in *AAAI Conference on Artificial Intelligence*, 2021. [Online]. Available: <https://api.semanticscholar.org/CorpusID:235306506>
- [34] U. Upadhyay, A. De, and M. Gomez Rodriguez, "Deep reinforcement learning of marked temporal point processes," *Advances in neural information processing systems*, vol. 31, 2018.
- [35] J. Sill, "Monotonic networks," *Advances in neural information processing systems*, vol. 10, 1997.
- [36] P. M. Chilinski and R. Silva, "Neural likelihoods via cumulative distribution functions," in *Conference on Uncertainty in Artificial Intelligence*, 2018. [Online]. Available: <https://api.semanticscholar.org/CorpusID:53294779>
- [37] D. Rindt, R. Hu, D. Steinsaltz, and D. Sejdinovic, "Survival regression with proper scoring rules and monotonic neural networks," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2022, pp. 1190–1205.
- [38] B. Liu, "Interpretable neural temporal point processes for modelling electronic health records," *ArXiv*, vol. abs/2404.08007, 2024.
- [39] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," in *NeurIPS 2014 Workshop on Deep Learning, December 2014*, 2014.
- [40] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.