Learning multivariate temporal point processes via the time-change theorem

Guilherme Augusto Zagatti
Institute of Data Science
National University of Singapore
Singapore

See-Kiong Ng
Institute of Data Science
and School of Computing
National University of Singapore
Singapore

Stéphane Bressan
School of Computing
and IPAL CNRS IRL2955
National University of Singapore
Singapore

Abstract

Marked temporal point processes (TPPs) are a class of stochastic processes that describe the occurrence of a countable number of marked events over continuous time. In machine learning, the most common representation of marked TPPs is the univariate TPP coupled with a conditional mark distribution. Alternatively, we can represent marked TPPs as a multivariate temporal point process in which we model each sequence of marks interdependently. We introduce a learning framework for multivariate TPPs leveraging recent progress on learning univariate TPPs via time-change theorems to propose a deep-learning, invertible model for the conditional intensity. We rely neither on Monte Carlo approximation for the compensator nor on thinning for sampling. Therefore, we have a generative model that can efficiently sample the next event given a history of past events. Our models show strong alignment between the percentiles of the distribution expected from theory and the empirical ones.

1 INTRODUCTION

Marked temporal point processes (TPPs) describe change happening at irregular random intervals. We can think of the realization of a marked TPP over continuous time as a sequence of data points recording the time an event occurred and any associated marks or attributes. The challenge with modelling TPPs is that we must explain not only the observed events but also the lack of events in between.

Marked TPPs can represent a plethora of real-world phenomena, which includes seismic shocks (Ogata, 1988), social interactions (Farajtabar et al., 2017), spiking activity in neural circuits (Linderman et al., 2017), electronic health records (Enguehard et al., 2020), and football league (Panos et al., 2023). In traditional applications, such as in the study of seismic shocks, compact TPP models such as the Hawkes process are derived from a combination of empirical observation, statistical analysis and physical modelling. In the past ten years, TPPs have entered the machine learning mainstream as neural TPPs, a family of models that leverage deep learning architectures to represent TPPs (Shchur et al., 2021).

Many neural TPPs rely on ad-hoc assumptions to propose a form for the conditional intensity. Alternatively, Shchur et al. (2020b) describes an approach grounded on the time-change theorem for univariate TPPs to design a new and efficient univariate neural TPPs. However, their method has not been extended to marked TPPs.

Most neural TPPs adopt the representation of marked TPPs as a ground, univariate TPP coupled with a mark distribution. Such representation often leads to the simplifying assumption that marks are conditionally independent of time given the learned history representation. While this simplification eases the computational burden, it limits the model's ability to capture the evolving relative likelihood of different marks over time. Existing neural TPPs attempting to break free from this constraint often suffer from efficiency issues in sampling.

In response to the above challenges, our paper presents a new contribution to neural TPPs. By restricting our focus to multivariate TPPs, we can re-cast the approach proposed by Shchur et al. (2020b) to the setting

Proceedings of the 27th International Conference on Artificial Intelligence and Statistics (AISTATS) 2024, Valencia, Spain. PMLR: Volume 238. Copyright 2024 by the author(s).

of marked TPPs. We use a version of the time-change theorem for multivariate TPPs to reveal a subtle, but important difference in how to approach data transformation with triangular maps. We propose three new model architectures for multivariate TPPs that take advantage of triangular maps. Unlike approaches that rely on Monte Carlo approximation to compute the compensator or thinning for sampling, triangular maps offer an elegant solution that avoids these drawbacks. Leveraging the theoretical foundations of TPPs, we also employ Q-Q plots to assess the goodness of fit of our trained model to the data, enhancing our understanding of neural TPPs in practice.

2 RELATED WORK

Early efforts of using TPPs to model real-world phenomena include the study of seismic shocks whose development is summarized in Vere-Jones (1995). Their account highlights the difficulty in developing a unified theory of earthquakes, and the rise of point processes as a tool to analyze seismologic data as a response. The Epidemic Type Aftershock-Sequences (ETAS), a type of Hawkes process with self-excitation, proved successful for the description of seismic catalogs, but the computation of its likelihood proved computationally demanding at the time (Ogata et al., 1993).

The advance of machine learning opened the doors to the estimation of models with a much larger number of parameters. Neural TPPs were developed partly as a response to Hawkes processes that are larger and more flexible than their predecessor. Du et al. (2016) pioneered neural TPPs with the recurrent marked TPP (RMTPP). They model the likelihood of the TPP as the product of the ground process and the conditional mark distribution. They assume the conditional independence between marks and time given the history of events. Their model first embeds marks and timestamps. Then, a recurrent neural network (RNN) encodes the history of past embedded events. A univariate TPP, whose conditional intensity is shaped according to a Gompertz distribution taking the encoded history and the current time as inputs, models the timestamps. The model uses the encoded history to predict marks using a logit-type layer.

As reviewed in Lin et al. (2021), different variations of this architecture came later including the event recurrent TPP (ERTPP) from Xiao et al. (2017) and the continuous time LSTM from Mei and Eisner (2017). The downside of these approaches is that they require Monte Carlo approximation to compute the compensator, which is the integrated conditional intensity, and is required for computing the log-likelihood as well as for event generation tasks. Event generation via thinning becomes inaccurate when coupled with Monte Carlo integration.

Instead of modelling TPPs via the conditional intensity, Shchur et al. (2020a) proposed modelling the conditional distribution of inter-event duration with the Log-Normal mixture (LogNorm) model. Although, the conditional intensity and the conditional distribution can be derived from one another, by directly modelling the conditional distribution, it is possible to sample the next event directly from it with no need for the compensator.

Later, Omi et al. (2019) proposed to model the compensator with the fully neural network intensity (FNNInt) model. The conditional intensity is then obtained via automatic differentiation. Shchur et al. (2020b) and Enguehard et al. (2020) proposed related approaches using triangular maps and attention respectively. Enguehard et al. (2020) denote these type of models cumulative-based and was the only one to extend these ideas to multivariate TPPs.

Some authors abandoned the conditional independence assumption between marks and time given the history of events. They modelled multivariate TPPs with separate conditional intensities for each mark using an approach inspired by Du et al. (2016). The Self-Attentive Neural Hawkes Process (*SAHP*) (Zuo et al., 2020), the Transformer Hawkes Process (*THP*) (Zhang et al., 2020) and Enguehard et al. (2020) use attention-based encoders for encoding the history. Attention proved more successful than RNN due to its capacity to attend to different parts of the history without any mediating representation that must travel through the modelled sequence. The difference between Zuo et al. (2020) and Zhang et al. (2020) boils down to the intensity function. Furthering this type approach, Zhang et al. (2021) assumes that multivariate TPPs are formed by sparsely connected TPPs, and incorporates a graph learning component to SAHP. All the models in this paragraph also rely on expensive Monte Carlo integration to compute the compensator.

More recently, we have observed new approaches to neural TPPs. Both Lin et al. (2022) and Dong et al. (2023) successfully explore generative-based neural TPPs. These methods also dwell on transforming random variables from their empirical form to well-understood representations. Lüdke et al. (2023) proposes a complete new approach based on the idea of thinning TPPs, but only study the univariate case.

In the theory of TPPs, the Poisson process has always played an important role akin to the Gaussian process in continuous processes. The Poisson process stands as the process with maximum entropy among all possible TPPs with a fixed mean rate. It is thus a remarkable result that marked TPPs that satisfy assumptions of time-change theorems can be transformed to a Poisson process (Daley and Vere-Jones, 2007). From a machine-learning perspective, these results suggest that any procedure for learning the parameters of an unknown TPP from data will be exhausted when it consistently transforms data to draws of a Poisson process. In this context, Shchur et al. (2020b) successfully designed models that satisfy the time-change theorem for univariate TPPs using a sequence of triangular maps. As far as we are aware, their approach has never been extended to marked TPPs, which is the topic of this paper.

3 FRAMEWORK

3.1 Revisiting the log-likelihood

We assume that an observer captures data over a fixed duration of time $[0, T)$, such that the n-th observation consists of the tuple (t_n, k_n). We denote the history of events up to but not including t as $H_{t-} = \{(t_n, k_n) \mid 0 \leq t_n < t\}$. Therefore, we denote the observed data as H_{T-}. The total number of observations N is a random variable. Marks are discrete. That is, the set of possible marks $\mathbb{K}$ is countable.

Any multivariate TPP can be characterized by its conditional intensity $\lambda^*(t, k) \equiv \lambda(t, k \mid H_{t-})$. The superscript $*$ denotes the conditioning of any function on the history of events up to but not including t. The conditional intensity is the probability that the next event will take place at time t with mark k given that no event has occurred since the last one in H_{t-}. Alternatively, the area under the conditional intensity within some interval is the number of expected points inside that interval.

Every multivariate TPP has a ground process, which is the sequence of timestamps without marks. The ground process is itself a univariate TPP. The conditional intensity of any multivariate TPP is factorizable in terms of the conditional intensity of the ground process $\lambda_g^*(t)$ and the conditional mark distribution. We assume the conditional mark distribution has a well-defined probability mass function $f_K^*(\cdot \mid t)$, where K denotes the random mark. As we will work with different probability mass functions, we sub-script them with the random variable they refer to. From the direct application of conditional probability, the conditional intensity function can be decomposed as $\lambda^*(t, k) = \lambda_g^*(t) f_K^*(k \mid t)$.

Let $\lambda_A^*(t) \equiv \lambda_g^*(t) \sum_{k \in A} f_K^*(k \mid t)$ and $\lambda_k^*(t) \equiv \lambda_{\{k\}}^*(t)$. We have that

$$\sum_{k \in A} f_K^*(k \mid t) = \frac{\lambda_A^*(t)}{\lambda_g^*(t)} \tag{1}$$

In other words, we can derive the conditional intensity for each mark if we know its conditional mark distribution and vice-versa. Many neural TPPs assume that $f_K^*(k \mid t)$ is independent of t given the learned history representation, which implies a constant likelihood between marks. For instance, see Lin et al. (2022).

Given $\lambda^*(t, k)$, its compensator is defined as $\Lambda^*(t, A) = \int_0^t \sum_{k \in A} \lambda^*(u, k)\, du$. Similar to the conditional intensity, let $\Lambda_A^*(t) \equiv \Lambda^*(t, A)$ and $\Lambda_k^*(t) \equiv \Lambda^*(t, \{k\})$. It follows that $\Lambda_{\mathbb{K}}^*(t) = \int_0^t \lambda_g^*(u)\, du = \Lambda_g^*(t)$.

The log-likelihood of the multivariate TPP is

$$\begin{aligned} \ell(H_{T^-}) &= \\ &= \sum_{n=1}^{N} (\log \lambda^*(t_n, k_n)) - \int_0^T \sum_{k \in \mathbb{K}} \lambda^*(u, k) du \\ &= \sum_{n=1}^{N} \left(\log \frac{\partial \Lambda_{k_n}^*}{\partial t_n}(t_n) \right) - \Lambda_g^*(T) \end{aligned} \tag{2}$$

The first part describes the log-likelihood of observing the recorded points, and the second part is the log-likelihood of not observing any point in between. See Chapter 7 Daley and Vere-Jones (2003) for more details.

Proposition 7.4.VI Daley and Vere-Jones (2003) state that a multivariate point process can be transformed into a multivariate Poisson process with independent components, each having unit rate via the compensator. We state a modified version of this theorem and propose a new proof based on Shchur et al. (2020b) that uses the change of random variable formula for probability densities. Steps of the proof serve as building blocks for the learning algorithm proposed in Subsection 3.2.

Theorem 1 (Random time change for multivariate TPP)**.** *Let $\mathbb{K}$ be the countable set of marks and H_{T^-} be the observed data from a multivariate TPP with σ-finite conditional intensity functions $\lambda_k^*(t) > 0$ for $k \in \mathbb{K}$. We assume that $\Lambda_k^*(t) \to \infty$ as $t \to \infty$. Let m_k the m-th event with mark k, such that $t_{m_{k_n}} = t_n$ is the time of the m-th event with mark k_n, where k_n is the mark of the n-th event in the data. The time transformation*

$$\Lambda^*(H_{T^-}) : (t_n, k_n) \mapsto \left(\tilde{t}_{m_{k_n}} \equiv \Lambda_{k_n}^*(t_n),\, k_n\right) \tag{3}$$

maps H_{T^-} into $|\mathbb{K}|$ independent sequences $\Lambda_k^*(H_{T^-}) \equiv (\tilde{t}_{m_{k_n}}, k_n | k_n = k)$ each of which is the realization of a univariate Poisson process with unit rate. Note that

$\Lambda_k^*(\cdot)$ *does not necessarily preserve the order of the original events between different sequences, that is,* $t_n < t_{n+1}$ *does not imply* $\tilde{t}_{m_{k_n}} < \tilde{t}_{m'_{k_{n+1}}}$.

Proof. First, we decompose the log-likelihood of any multivariate TPP into the log-likelihood of each component.

$$\begin{aligned}\ell(H_{T^-}) &= \sum_{n=1}^{N} \log \lambda^*(t_n, k_n) - \int_0^T \sum_{k\in\mathbb{K}} \lambda^*(u,k)du \\ &= \sum_{n=1}^{N} \log \lambda_g^*(t) f_K^*(k_n \mid t) - \int_0^T \lambda_g^*(t) \sum_{k\in\mathbb{K}} f_K^*(k \mid t)du \\ &= \sum_{k\in\mathbb{K}} \left[\sum_{m_k=1_k}^{N_k} \log \lambda_k^*\left(t_{m_k}\right) - \Lambda_k^*(t) \right]\end{aligned} \tag{4}$$

The exchange of the integral with the sum from the second to the third line follows from Tonelli's theorem.

Next, we write our proposed transformation $\Lambda^*(H_T^-)$ in vector format. Read t_{1_k} and t_{N_k} as the first and last event with mark k respectively.

$$\Lambda^*(H_{T^-}) = \begin{bmatrix} \Lambda_1^*(H_{T^-}) \\ \vdots \\ \Lambda_{|\mathbb{K}|}^*(H_{T^-}) \end{bmatrix} = \begin{bmatrix} \Lambda_1^*(t_{1_1}) \\ \vdots \\ \Lambda_1^*(t_{N_1}) \\ \vdots \\ \Lambda_{|\mathbb{K}|}^*(t_{1_{|\mathbb{K}|}}) \\ \vdots \\ \Lambda_{|\mathbb{K}|}^*(t_{N_{|\mathbb{K}|}}) \end{bmatrix} \tag{5}$$

From its definition, the compensator is a function of t_n and $H_{t_n^-}$, therefore we have that $\partial\Lambda_k^*/\partial t_n(t_n) = \lambda_k^*(t_n)$, and, for $m > n$, $\partial\Lambda_k^*/\partial t_m(t_n) = 0$. Thus, we obtain the Jacobian described in Supplement Equation 1.

The Jacobian of our proposed transformation has interesting properties. Its diagonal is equal to the conditional intensity evaluated at each event. The off-diagonal entries are zero whenever $t_{n_k} < t_{m_l}$. Therefore, if we sort the rows of the compensator according to the time of the events, the Jacobian becomes a triangular matrix, which implies that its determinant is equal to the product of its diagonal entries.

$$\begin{aligned}|J_{\Lambda^*}(H_{T^-})| &= \prod_{n=1}^{N} \lambda_{k_n}^*(t_n) \\ &= \prod_{k\in\mathbb{K}} \prod_{m_k=1_k}^{N_k} \lambda_k^*\left(t_{m_k}\right) \\ &= \prod_{k\in\mathbb{K}} |J_{\Lambda_k^*}(H_{T^-})|\end{aligned} \tag{6}$$

The change of random variable formula for probability densities states the following. Let two random variables X and Y defined on the same probability space with probability densities f_X and f_Y respectively such that $Y = G(X)$, G is bijective and differentiable and J_G is the Jacobian of G, then:

$$Y = G(X) \iff \log f_X(x) = \log|J_G(x)| + \log f_Y\left(G(x)\right) \tag{7}$$

Let $X = H_{T^-}$ and $Y = \Lambda^*(H_{T^-})$. From Equation 6, we obtain

$$\log|J_G(x)| = \log|J_{\Lambda^*}(H_{T^-})| = \sum_{k\in\mathbb{K}} \sum_{m_k=1_k}^{N_k} \log \lambda_k^*\left(t_{m_k}\right) \tag{8}$$

If we assume that Y consists of $|\mathbb{K}|$ independent Poisson processes with unit rate, its log-likelihood is equal to $N = \sum_{k\in\mathbb{K}} N_k$ draws from the exponential distribution.

$$\begin{aligned}\log f_Y(y) &= \ell(\Lambda^*(H_{T^-})) \\ &= \sum_{k\in\mathbb{K}} \left(\sum_{m_k=1_k}^{N_k} \log 1 - \int_0^{\Lambda_k^*(T)} 1\ du \right) \\ &= \sum_{k\in\mathbb{K}} -\Lambda_k^*(T)\end{aligned} \tag{9}$$

Adding Equation 8 and 9, we obtain Equation 4 which is the log-likelihood of the original TPP. Therefore, the proposed transformation $\Lambda^*(H_{T^-})$ transforms the multivariate TPP into $|\mathbb{K}|$ independent Poisson processes with unit rate. ☐

3.2 Normalizing flows for multivariate TPPs

Normalizing flows allow the expression of complex distributions with simple ones which can be useful for learning and generation tasks as reviewed in Papamakarios et al. (2021). Let X be a random variable which we want to model, the defining element of the normalizing flow is the bijective and differentiable transformation G that maps X to a random variable with a simpler distribution Y. While Y can be easily sampled and/or fit to data, X cannot.

As we saw in Subsection 3.1, the compensator of a multivariate TPP is a bijective transformation that maps a complex, unknown distribution to a simple, well-known distribution, namely the exponential distribution. In Shchur et al. (2020b), the authors propose an architecture based on normalizing flow for learning univariate TPPs via triangular maps. A triangular map $G : \mathbb{R}^N \to \mathbb{R}^N$ is composed of N functions such that each component function g_n depends only on the first n elements of its domain $(x_1, \cdots, x_n)$ and is increasing on x_n. The previous Subsection shows that the compensator of a multivariate TPP is indeed a triangular map. Alternatively, any triangular map G that satisfies the conditions of Theorem 1 is a valid compensator. Like with the compensator, we denote $G_k(H_{T^-}) \equiv ((G_k(t_n) \equiv G(t_n, k_n), k_n \mid k_n = k))$. Then, G is a valid compensator, if each G_k is bijective on the space of increasing sequences restricted to k. Note the subtle, but important difference between the condition stated here and Condition 2 of Section C.3 of Shchur et al. (2020b). In our case G only needs to preserve the order of elements in the sequence with the same mark, while in Shchur et al. (2020b) the order of elements should be preserved in the whole sequence.

3.3 Learning triangular maps

We model the data H_{T^-} using a parameterized triangular map $G^*(\cdot \mid \theta)$ by minimizing the average negative log-likelihood via gradient descent. Let $g^*(t_n) \equiv \partial G^*/\partial t_n(t_n)$ and re-writing Equation 4 in terms of G we obtain our minimization problem.

$$\hat{\theta} = \min_{\theta} \frac{1}{N} \sum_{k \in \mathbb{K}} \left[\sum_{m_k = 1_k}^{N_k} \log g_k^* \left(t_{m_k} \mid \theta\right) - G_k^*(T \mid \theta) \right] \tag{10}$$

Borrowing the lessons from Shchur et al. (2020b) we propose to construct G as a sequence of triangular maps. However, since our discussion in Subsection 3.2 reveals that G only needs to preserve the order of the transformation within each mark k, we modify the layers introduced by Shchur et al. (2020b) to increase the expressiveness of our model. First, we modify the affine layer A to use separate scaling and bias parameters for each mark k. Next, the pairwise difference matrix D depicted in Supplement Figure 1 applies the pairwise difference between consecutive timestamps of the same mark, that is, for $n = m_{k_n}$, $D(t_n) = t_{m_{k_n}} - t_{(m-1)_{k_n}}$ which is performed in $\mathcal{O}(N|\mathbb{K}|)$ steps. Cumulative sum $C \equiv D^{-1}$ performs the reverse operation in the same number of steps. Shchur et al. (2020b) borrows the rational quadratic spline function from Durkan et al. (2019) as a flexible operator with well-defined derivatives and inverse. We modify the spline layer Φ such that we train a set of spline parameters for each mark k. Given t_n we apply $\Phi_{k_n}(t_n)$, see Supplement Figure 2. To capture the dependency of events on previous events, Shchur et al. (2020b) proposes the use of a sequence of block-diagonal matrices B_l, "where each block is a repeated $H \times H$ lower-triangular matrix with strictly positive diagonal entries. [...] The blocks in every other layer are shifted by an offset $H/2$." Again, we propose to train a block-diagonal matrix for each mark k as illustrated in Supplement Figure 4. To do that, for every mark k in the model we multiply all the timestamps by $B_{l,k}$. We allocate the transformed timestamps by masking with k. If multiplication by the original B_l takes $\mathcal{O}(NH)$, then the marked version of B_l takes $\mathcal{O}(|\mathbb{K}|NH)$. Similarly, computing the inverse $B_{l,k}^{-1}$ takes $\mathcal{O}(H^2)$, so the inverse of B_l^{-1} takes $\mathcal{O}(|\mathbb{K}|H^2)$. Since transformations are sequential, the transformation of timestamp with mark k at layer l might impact the transformation of timestamp with mark k' in layer $l+1$ which means that, even though we apply mark-specific parameters at each layer, these mark-specific transformations are transmitted down the chain to other marks.

In the light of our discussion, we re-assess the SAHP model by Zhang et al. (2020). SAHP innovates in separately modelling the conditional intensity of each mark like we propose in this paper. Their model is also a triangular map since they use a causal attention layer to encode the history. However, their model suffers the drawback that it computes the compensator via Monte Carlo. It would require modifications to become invertible and, even so, it would need root finding methods for inversion [1]. To overcome these challenges, we propose an alternative attention-based triangular map layer R. We encode H_{T^-} using a causal attention layer following Zhang et al. (2020). Using a linear layer, we transform the encoded history of dimension E to the parameters of a spline layer to obtain a vector of order $\mathcal{O}(|\mathbb{K}|)$. This is very similar to Zhang et al. (2020), who transform the encoded history to the parameters of their Hawkes model. The benefit of our approach is that the spline can be inverted as easily as the forward pass which is shown in Supplement Figure 4. The disadvantage is that the spline has a much larger number of parameters. This approach is also inspired by Shchur et al. (2020b) implementation of RNN using triangular maps.

Using our modified layers, we train three models adapted from Shchur et al. (2020b): the multivariate modulated renewal (*MultiMRP*), $G = C \circ \Phi_1 \circ D \circ A \circ \Phi_0$, *MultiTriTPP*, $G = C \circ \Phi_2 \circ B_L \circ \cdots \circ B_1 \cdots \Phi_1 \circ D \circ A \circ \Phi_0$, and the multivariate spline transformer (*MultiTraTPP*), $G = C \circ R \circ D \circ A \circ \Phi_0$. Following Lin et al. (2022), we re-scale the data with a fixed affine transformation by dividing each timestamp by the maximum timestamp observed in the training set and multiplying them by 50. If we denote this transformation as A_0, this is equivalent to $G \circ A_0$. This helps to stabilize training. See Supplement Section 2 for model diagrams and more details.

To train multivariate TPPs using triangular maps we must supply as input, in addition to H_{T^-}, a $|\mathbb{K}|$-dimensional vector with repeated entries containing

[1]As the vanilla Hawkes process does.

	QQD (↓)						MAPE (↓)					
	Yelp	*Hawkes*	*Retweet*	*SO*	*MIMIC*	*MOOC*	*Yelp*	*Hawkes*	*Retweet*	*SO*	*MIMIC*	*MOOC*
MultiMRP	0.14	**0.01**	0.07	0.08	0.87	0.31	15.89	6.24	13.11	6.83	1.29	65.69
MultiTriTPP	0.08	**0.01**	0.04	0.06	0.70	0.38	16.89	**1.24**	7.81	**1.51**	2.08	9.07
MultiTraTPP	**0.05**	0.04	0.04	**0.04**	0.71	0.27	14.96	4.52	9.09	4.59	3.36	68.25
DETER	39.89	30.98	29.99	32.96	26.87	8.53	14.72	5.61	21.47	7.01	3.21	69.89
RMTPP	0.25	0.48	0.65	0.93	0.52	0.97	42.29	11.22	69.59	34.99	3.57	87.56
LogNorm	0.24	0.48	0.66	0.93	0.97	0.97	32.94	52.18	86.03	49.29	21.48	94.26
ERTPP	0.38	0.48	0.69	0.93	0.38	0.97	44.61	38.93	95.37	49.30	18.90	94.26
WeibMix	0.98	0.98	0.98	0.98	0.98	0.98	74.84	28.10	62.27	49.30	21.43	93.96
FNNInt	0.24	0.05	0.11	0.06	1.97	0.45	15.55	5.42	5.47	6.72	1.49	17.11
SAHP	18.79	2.18	284.20	2.20	0.64	413.62	13.36	5.27	**2.46**	5.02	0.75	**1.95**
THP	10.59	2.88	15.15	0.66	0.83	15.04	17.09	5.23	16.06	5.73	**0.75**	42.30
TCDDM	0.51	0.03	**0.01**	0.16	**0.37**	0.10	74.84	7.29	17.02	19.00	21.47	72.27
TCVAE	13.85	1.19	0.13	2.19	6.24	**0.07**	**10.81**	4.66	9.55	6.13	1.47	75.22
TCGAN	53.82	42.87	17.24	1.52	49.55	46.85	11.75	4.20	40.39	2.26	1.46	8.13
TCCNF	0.43	0.10	0.25	0.15	5.28	0.95	19.13	6.71	22.79	9.63	0.91	90.98
TCNSN	0.51	0.46	0.63	0.47	0.62	0.74	N/A	57.40	95.37	49.30	21.47	94.26

Table 1: Test set model evaluation. QQ deviation (*QQD*) and mean absolute percentage error (*MAPE*). Best values are bold and second-best are underlined.

the last timestamp of the sequence $(T, \cdots, T)$. This vector will be used to compute the compensated timestamp at the end of the observation time T for each mark as required by Equation 10. This is a unique requirement for the multivariate models we are proposing, since the last timestamp of H_{T^-} is only transformed according to its associated mark k_N. Therefore, we need the additional information to compute the correct log-likelihood.

3.4 Sampling triangular maps

Despite methods that improve sampling efficiency, marked TPP sampling is inherently sequential as reviewed in Zagatti et al. (2023). The reason is that the history up to the beginning of the sampling interval is required in order to compute the conditional distribution, which in turn is required for sampling beyond that point. This is true even when sampling the ground process because its conditional distribution can depend on the mark history. There are two main methods for sampling marked TPPs either via inverting the compensator or via thinning. Thinning requires knowledge of the future path of the conditional intensity in order to determine its upper bound. An inefficient upper bound can lead to high-rejection rates and less efficient sampling. Alternatively, inverting the compensator allows for fast and rejection-free sampling if we can efficiently and accurately perform this computation. Since our models are based on triangular maps, they can be easily inverted as discussed in Section 3.3.

Taking advantage of this capability, we implement the first reaction sampling method for sampling the next event (Zagatti et al., 2023). The method proceed as following. First, we push H_{T^-} and $(T, \cdots, T)$ through our model to obtain $\tilde{H}_{T^-}$ and $(\tilde{T}_1, \cdots, \tilde{T}_K)$. For each mark k, we sample a value from the exponential distribution with unit rate conditional on $\tilde{T}_K$. Since the exponential distribution has the memoryless property this amounts to sampling from the exponential distribution and adding to $\tilde{T}_k$. We obtain the vector of proposals for the next event time $(\tilde{\tilde{T}}_1, \cdots, \tilde{\tilde{T}}_K)$ in the compensated space. We invert the proposals back to the original temporal space using $\tilde{H}_{T^-}$ to obtain the sampled time for each mark. The earliest timestamp $(\hat{T}_1, \cdots, \hat{T}_k)$ and its associated mark are selected as the next sampled event. It is possible to sample multiple next events in parallel. However, sampling an entire sequence can only be done sequentially.

4 EXPERIMENTS

4.1 Objectives and data

The main advantage of our model is to learn TPPs that satisfy Theorem 1, so we adopt as our main evaluation metric the mean absolute deviation between the expected and computed percentiles (*QQD*). The objective of our experiments is to evaluate whether our proposed framework remains competitive against al-

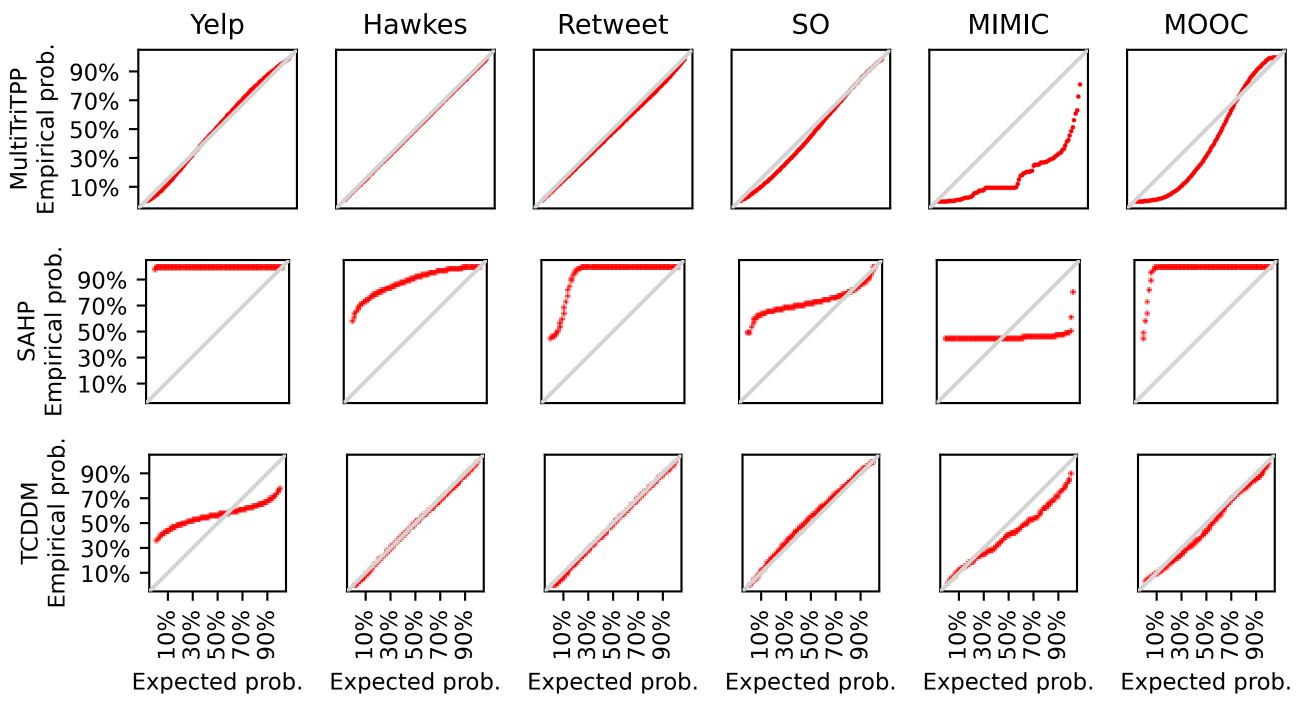

Figure 1: Q-Q plot: expected versus predicted quantiles for ground process.

ternatives proposed in the literature.

We also benchmark the mean absolute percentage error (*MAPE*) between the predicted and expected next event time, next event mark top-1 accuracy (*TOP1*) and top-3 accuracy (*TOP3*). When computing metrics, we consider all data points in the sequences tested. See Supplement Section 4 for more computational details about all the metrics proposed.

The benchmark models are the same selected by Lin et al. (2022). We consider a baseline deterministic model (*DETER*), most of the models reviewed in Section 2, namely *RMTPP*, *LogNorm*, *ERTPP*, *FNNInt*, *SAHP*, *THP*, and the temporal conditional models from Lin et al. (2022), diffusion denoising (*TCDDM*), variational autoencoder (*TCVAE*), generative adversarial network (*TCGAN*), continuous normalizing flows (*TCCNF*), noise score network (*TCNSN*). See Supplement Section 5 for a detailed description.

We benchmark six datasets. The first dataset is the synthetic dataset described in Zhang et al. (2020), which consists of the simulation of a multivariate Hawkes process with exotic kernels (*Hawkes*). The remaining five datasets come from real-world phenomena: *Yelp* consists of sequences of restaurant check-ins without any marks; *Retweet* (Zhao et al., 2015) consists of re-tweet sequences classified into three categories based on user popularity; *SO* (Du et al., 2016) consists of sequences of 22 possible user awards over two years; *MIMIC* (Lee et al., 2011) consists of sequences of 75 possible diagnosis during clinical visits over seven years; *MOOC* (Kumar et al., 2019) consists of sequences of 97 possible different user interactions with an on-line course system. We use *Hawkes*, *Retweet* and *MIMIC* made available by Zhang et al. (2020), and *Yelp*, *StackOverflow* and *MOOC* by Lin et al. (2022).

We base our benchmark suite on the comprehensive battery of tests from Lin et al. (2022). However, we noticed issues with their implementation of *QQD*. We correct their computation and re-train all their models with their default settings on the datasets listed above [2]. Apart from *QQD*, we mostly get similar results.

4.2 Implementation details

Our proposed framework is implemented with Torch 2.0, and we borrow source code from Shchur et al. (2020b) and Lin et al. (2022). The source code is freely available [3]. To run our experiments, we used a computer running Intel Xeon E5 v4 2.20 GHz with 40 CPU cores, 500 GB of memory, 8 Nvidia 32Gb V100 GPUs and CUDA version 12. We tune the following hyperparameters: number of spline knots $\{10, 20, 50\}$,

[2]Our fork is available at `https://github.com/gzagatti/GNTPP`.

[3]`https://github.com/NUS-IDS/multi-ttpp`

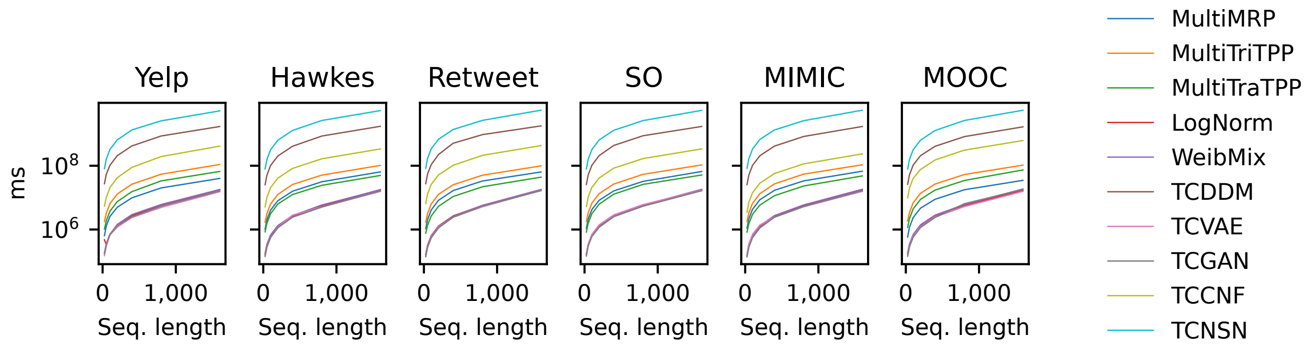

Figure 2: Trajectory sampling performance for neural TPP models

number of blocks $\{2, 4\}$, block size $\{8, 16, 32\}$ and embedding size $\{8, 16\}$. The model that obtains the best *QQD* on the validation set is selected. We also experimented with selecting hyperparameters based on best negative log-likelihood but found models to be more robust when using *QQD*. The training routine uses the `AdamW` optimizer, the `ReduceLROnPlateau` learning rate scheduler, and early stopping with patience set to 15.

4.3 Results and discussion

Table 1 summarize our main results. Regarding *QQD*, all three models we propose obtain the best values for datasets with small to medium mark dimensionality. They become less competitive as the mark dimensionality grows. Our models perform the worst against the *MIMIC* dataset. When facing *MOOC*, which has the biggest mark dimensionality, our models recover some of their edge but become less competitive with other alternatives. *MIMIC* contains numerous marks which appear only sparsely. Some of the marks do not even show up in the training sets. Although our models are equipped to model the non-appearance of marks, it does not perform as well. Theorem 1 assumes that the conditional intensities for all marks are strictly positive, which is unlikely for *MIMIC*. This assumption is equivalent to assuming no last points in the process. One could potentially adapt the theorem for such cases as Daley and Vere-Jones (2003) Example 7.4 (b) suggests, but it is out of the scope of this paper. *Hawkes*, *Retweet* and *SO* are datasets in which all marks can appear at any moment leading to a strong fit. In addition to that, the validation and test sets of *MIMIC* are small, increasing the variance of the estimated empirical quantiles required to compute *QQD*.

When we analyze Figure 1 which depicts Q-Q plots of ground process for selected models [4], we can more clearly observe the capacity of our models to learn the correct compensator for the data. The Q-Q plots for *MultiTriTPP* show that, indeed, the compensated data is very close to the exponential distribution for four out of six datasets. The *MIMIC* is the most problematic dataset. The diffusion-based *TCDDM* also obtains strong *QQD* performance, which is reflected in the Q-Q plots. However, *TCDDM* performs less well with the *Yelp* dataset, which although contains no marks, is the most challenging in terms of sequence lengths with an average of 580 check-ins per sequence, more than four times longer than the next dataset. In general, *SAHP* achieves strong results with predictive marks. However, its Q-Q plots show that it has problems in finding the compensated form of the data. This shortcoming reflected in its *QQD* score is indicative of potential shortcomings in its use as a TTP. Therefore, additional investigation is required. Alternatively, we can use Q-Q plots to evaluate the goodness-of-fit of models to data. The poor fit of the *MIMIC* dataset in addition to the problems highlighted above suggest that TPP models might not be a good modelling approach for it.

To investigate efficiency in generative tasks, we conducted a benchmark exercise in which we measured the time it took to generate sequences from 50 to $1,600$ events for 10 sequences in parallel. Results are depicted in Figure 2. *SAHP* and *THP* do not support sampling — see Lin et al. (2022). *RMTPP*, *ERTPP*, *FNNInt* and *DETER* can sample negative time intervals, which means that they do not produce correct trajectories [5], so they were not benchmarked. Our methods rank in the middle, scaling linearly with sequence length as expected from 3.4. While not as efficient as models that sample directly from the inter-event time distribution, it is faster than all models that sample from the conditional intensity.

[4]See Supplement Figure 5 for disaggregated processes.

[5]At least with the traditional TPP sampling algorithms implemented in the source code.

Next, it is informative to consider the shape of the conditional intensity. For the Hawkes process, we can analytically compute it, and compare it with our fitted models. For this discussion, we pick a random sequence from the Hawkes test set and plot the intensity for the first mark. We plot two models in Figure 3. Both models follow the general path of the true intensity. *MultiTriTPP* is characterized by spikes of a much larger magnitude than that observed in data. *SAHP* tends to stay above the true intensity. It has smooth decay as it is modelled after a Hawkes process with an exponential kernel. Overall, none of the benchmarked models have a perfect fit [6].

The performance of the models on prediction-focused metrics does not always agree with *QQD*. Starting with *MAPE* we note that our models offer good performance. *MultiTriTPP* obtains the best *MAPE* on the *Hawkes* and *SO* datasets. Again, we see some deterioration in performance when mark dimensionality increases and marks become more sparse. Regarding other models, there are disagreements between *MAPE* and *QQD*. Some models like *SAHP*, *THP* and *TCGAN* have strong *MAPE* but weak *QQD*. Alternatively, *TCDDM* has strong *QQD* and weak *MAPE*, which is also encountered by Lüdke et al. (2023).

Next, we list mark accuracy metrics in Supplement Table 3. On the one hand, our models tend to perform poorly in these metrics, which is surprising since they are designed with mark interdependency in mind. On the other hand, all the models tested in Lin et al. (2022) have strong, but similar mark accuracy as they use a common approach for modelling marks. Enguehard et al. (2020) also find that their cumulative-based models are harder to train, display accuracy metrics with high variance, and mismatch between the relative performance of the negative log-likelihood and accuracy. Alternatively, their most performant models are their Monte Carlo based ones that are closer to *SAHP*. Unfortunately, they do not evaluate *QQD* or *MAPE* in their work.

As an alternative to *SAHP*, we proposed *MultiTraTPP*. Both models encode the history of events using attention, but differ on the parameterization of the conditional intensity. Since the rational-quadratic splines used to parameterize the conditional intensity of *MultiTraTPP* are flexible enough to represent a wide range of functions (Durkan et al., 2019). Surprisingly, *MultiTraTPP* fails to obtain the strong performance of *SAHP* in predictive-focused metrics.

Finally, a characteristic of our models that shows up in our experiment is the fact that the total number of parameters grows with the dimensionality of the marks at rate $\mathcal{O}(|\mathbb{K}|)$. While our model is compact when dealing with small to medium mark dimensionality, it grows significantly larger with bigger dimensionality. This also affects other models like *SAHP* and *THP* that use the multivariate representation of TTPs as we do, but to a lower extent.

[6] See Supplement Figure 7 for more.

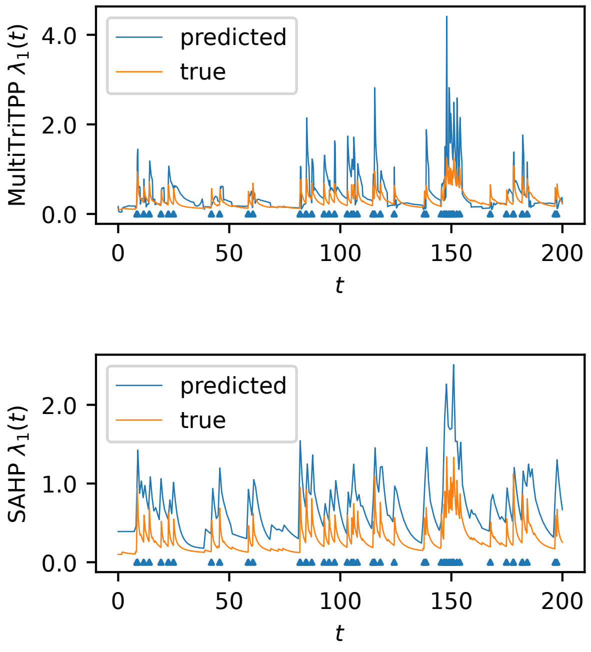

Figure 3: Predicted versus estimated intensity rate, Hawkes process. True sequence at bottom.

5 CONCLUSION

In this paper, we used the time-change theorem for multivariate TPPs to extend the work of Shchur et al. (2020b) to the setting of marked TPPs. The proposed framework uses triangular maps to design neural TPPs that rely neither on Monte Carlo approximation to compute the compensator or thinning for sampling. We conducted extensive experiments to demonstrate that our models can find the compensated form of the data whose empirical distribution closely tracks the exponential distribution as expected by theory. In the future, we would like to investigate attention-based triangular maps for stronger predictive performance. It remains unsolved the gap between *QQD* or *MAPE* and accuracy metrics for cumulative-based neural TPPs. Alternatively, research should focus on reducing the rate at which the number of parameters of the proposed triangular maps scale with mark dimensionality. One option is to model TPPs as a network of sparsely connected TPPs as proposed in Zhang et al. (2021).

References

Daley, D. J. and Vere-Jones, D. (2003). *An Introduction to the Theory of Point Processes: Volume I: Elementary Theory and Methods*. Probability and Its Applications, An Introduction to the Theory of Point Processes. Springer-Verlag, New York, 2 edition.

Daley, D. J. and Vere-Jones, D. (2007). *An Introduction to the Theory of Point Processes: Volume II: General Theory and Structure*. Springer, New York, 2nd edition edition.

Dong, Z., Fan, Z., and Zhu, S. (2023). Conditional Generative Modeling is All You Need for Marked Temporal Point Processes.

Du, N., Dai, H., Trivedi, R., Upadhyay, U., Gomez-Rodriguez, M., and Song, L. (2016). Recurrent Marked Temporal Point Processes: Embedding Event History to Vector. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, New York, NY, USA. Association for Computing Machinery.

Durkan, C., Bekasov, A., Murray, I., and Papamakarios, G. (2019). Neural Spline Flows.

Enguehard, J., Busbridge, D., Bozson, A., Woodcock, C., and Hammerla, N. (2020). Neural Temporal Point Processes For Modelling Electronic Health Records. In *Proceedings of the Machine Learning for Health NeurIPS Workshop*. PMLR.

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

Kumar, S., Zhang, X., and Leskovec, J. (2019). Predicting Dynamic Embedding Trajectory in Temporal Interaction Networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, Anchorage AK USA. ACM.

Lee, J., Scott, D. J., Villarroel, M., Clifford, G. D., Saeed, M., and Mark, R. G. (2011). Open-access MIMIC-II database for intensive care research. In *2011 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*.

Lin, H., Tan, C., Wu, L., Gao, Z., and Li, S. Z. (2021). An Empirical Study: Extensive Deep Temporal Point Process.

Lin, H., Wu, L., Zhao, G., Pai, L., and Li, S. Z. (2022). Exploring Generative Neural Temporal Point Process. *Transactions on Machine Learning Research*.

Linderman, S. W., Wang, Y., and Blei, D. M. (2017). Bayesian Inference for Latent Hawkes Processes. In *Conference on Neural Information Processing Systems (NIPS 2017)*.

Lüdke, D., Biloš, M., Shchur, O., Lienen, M., and Günnemann, S. (2023). Add and Thin: Diffusion for Temporal Point Processes. In *Thirty-Seventh Conference on Neural Information Processing Systems*.

Mei, H. and Eisner, J. M. (2017). The Neural Hawkes Process: A Neurally Self-Modulating Multivariate Point Process. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Ogata, Y. (1988). Statistical Models for Earthquake Occurrences and Residual Analysis for Point Processes. *Journal of the American Statistical Association*, 83(401).

Ogata, Y., Matsu'ura, R. S., and Katsura, K. (1993). Fast likelihood computation of epidemic type aftershock-sequence model. *Geophysical Research Letters*, 20(19).

Omi, T., Ueda, N., and Aihara, K. (2019). Fully neural network based model for general temporal point processes. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, number 190. Curran Associates Inc., Red Hook, NY, USA.

Panos, A., Kosmidis, I., and Dellaportas, P. (2023). Scalable marked point processes for exchangeable and non-exchangeable event sequences. In *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*. PMLR.

Papamakarios, G., Nalisnick, E., Rezende, D. J., Mohamed, S., and Lakshminarayanan, B. (2021). Normalizing Flows for Probabilistic Modeling and Inference.

Shchur, O., Biloš, M., and Günnemann, S. (2020a). Intensity-free learning of temporal point processes. *arXiv:1909.12127 [cs, stat]*.

Shchur, O., Gao, N., Biloš, M., and Günnemann, S. (2020b). Fast and Flexible Temporal Point Processes with Triangular Maps. In *Advances in Neural Information Processing Systems*, volume 33. Curran Associates, Inc.

Shchur, O., Türkmen, A. C., Januschowski, T., and Günnemann, S. (2021). Neural Temporal Point Processes: A Review. In *Twenty-Ninth International Joint Conference on Artificial Intelligence*, volume 5.

Vere-Jones, D. (1995). Forecasting earthquakes and earthquake risk. *International Journal of Forecasting*, 11(4).

Xiao, S., Yan, J., Chu, S. M., Yang, X., and Zha, H. (2017). Modeling The Intensity Function Of Point Process Via Recurrent Neural Networks.

Zagatti, G. A., Isaacson, S. A., Rackauckas, C., Ilin, V., Ng, S.-K., and Bressan, S. (2023). Extending JumpProcess.jl for fast point process simulation with time-varying intensities.

Zhang, Q., Lipani, A., Kirnap, O., and Yilmaz, E. (2020). Self-Attentive Hawkes Processes. *arXiv:1907.07561 [cs, stat]*.

Zhang, Q., Lipani, A., and Yilmaz, E. (2021). Learning Neural Point Processes with Latent Graphs. In *Proceedings of the Web Conference 2021*, WWW '21, New York, NY, USA. Association for Computing Machinery.

Zhao, Q., Erdogdu, M. A., He, H. Y., Rajaraman, A., and Leskovec, J. (2015). SEISMIC: A Self-Exciting Point Process Model for Predicting Tweet Popularity. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, New York, NY, USA. Association for Computing Machinery.

Zuo, S., Jiang, H., Li, Z., Zhao, T., and Zha, H. (2020). Transformer Hawkes Process. In *Proceedings of the 37th International Conference on Machine Learning*. PMLR.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes]
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes]
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Yes]

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Yes]
   (b) Complete proofs of all theoretical results. [Yes]
   (c) Clear explanations of any assumptions. [Yes]

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes]
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes]
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes]
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Yes]
   (b) The license information of the assets, if applicable. [Yes]
   (c) New assets either in the supplemental material or as a URL, if applicable. [Yes]
   (d) Information about consent from data providers/curators. [Not Applicable]
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Not Applicable]
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

Learning multivariate temporal point processes via the time-change theorem: Supplementary Materials

1 JACOBIAN

The Jacobian of the compensator J_{Λ^*} can be transformed into a triangular matrix by re-arranging the rows, as we can see from its expansion below.

$$
\begin{aligned}
&J_{\Lambda^*}(H_{T^-}) = \\
&= \begin{bmatrix}
\frac{\partial \Lambda_1^*}{\partial t_{1_1}}(t_{1_1}) & \cdots & \frac{\partial \Lambda_1^*}{\partial t_{N_1}}(t_{1_1}) & \cdots & \frac{\partial \Lambda_1^*}{\partial t_{1_{|\mathbb{K}|}}}(t_{1_1}) & \cdots & \frac{\partial \Lambda_1^*}{\partial t_{N_{|\mathbb{K}|}}}(t_{1_1}) \\
 & & & \vdots & & & \\
\frac{\partial \Lambda_1^*}{\partial t_{1_1}}(t_{N_1}) & \cdots & \frac{\partial \Lambda_1^*}{\partial t_{N_1}}(t_{N_1}) & \cdots & \frac{\partial \Lambda_1^*}{\partial t_{1_{|\mathbb{K}|}}}(t_{N_1}) & \cdots & \frac{\partial \Lambda_1^*}{\partial t_{N_{|\mathbb{K}|}}}(t_{N_1}) \\
 & & & \vdots & & & \\
\frac{\partial \Lambda_K^*}{\partial t_{1_1}}(t_{1_{|\mathbb{K}|}}) & \cdots & \frac{\partial \Lambda_K^*}{\partial t_{N_1}}(t_{1_{|\mathbb{K}|}}) & \cdots & \frac{\partial \Lambda_K^*}{\partial t_{1_{|\mathbb{K}|}}}(t_{1_{|\mathbb{K}|}}) & \cdots & \frac{\partial \Lambda_K^*}{\partial t_{N_{|\mathbb{K}|}}}(t_{1_{|\mathbb{K}|}}) \\
 & & & \vdots & & & \\
\frac{\partial \Lambda_K^*}{\partial t_{1_1}}(t_{N_{|\mathbb{K}|}}) & \cdots & \frac{\partial \Lambda_K^*}{\partial t_{N_1}}(t_{N_{|\mathbb{K}|}}) & \cdots & \frac{\partial \Lambda_K^*}{\partial t_{1_{|\mathbb{K}|}}}(t_{N_{|\mathbb{K}|}}) & \cdots & \frac{\partial \Lambda_K^*}{\partial t_{N_{|\mathbb{K}|}}}(t_{N_{|\mathbb{K}|}})
\end{bmatrix} \qquad (1) \\
&= \begin{bmatrix}
\lambda_1^*(t_{1_1}) & \cdots & 0 & \cdots & \mathbb{1}(t_{1_1} > t_{1_{|\mathbb{K}|}})\frac{\partial \Lambda_1^*}{\partial t_{1_{|\mathbb{K}|}}}(t_{1_1}) & \cdots & \mathbb{1}(t_{1_1} > t_{N_{|\mathbb{K}|}})\frac{\partial \Lambda_1^*}{\partial t_{N_{|\mathbb{K}|}}}(t_{1_1}) \\
 & & & \vdots & & & \\
\frac{\partial \Lambda_1^*}{\partial t_{1_1}}(t_{N_1}) & \cdots & \lambda_1^*(t_{N_1}) & \cdots & \mathbb{1}(t_{N_1} > t_{1_{|\mathbb{K}|}})\frac{\partial \Lambda_1^*}{\partial t_{1_{|\mathbb{K}|}}}(t_{N_1}) & \cdots & \mathbb{1}(t_{N_1} > t_{N_{|\mathbb{K}|}})\frac{\partial \Lambda_1^*}{\partial t_{N_{|\mathbb{K}|}}}(t_{N_1}) \\
 & & & \vdots & & & \\
\mathbb{1}(t_{1_{|\mathbb{K}|}} > t_{1_1})\frac{\partial \Lambda_K^*}{\partial t_{1_1}}(t_{1_{|\mathbb{K}|}}) & \cdots & \mathbb{1}(t_{1_{|\mathbb{K}|}} > t_{N_1})\frac{\partial \Lambda_K^*}{\partial t_{N_1}}(t_{1_{|\mathbb{K}|}}) & \cdots & \lambda_K^*(t_{1_{|\mathbb{K}|}}) & \cdots & 0 \\
 & & & \vdots & & & \\
\mathbb{1}(t_{N_{|\mathbb{K}|}} > t_{1_1})\frac{\partial \Lambda_K^*}{\partial t_{1_1}}(t_{N_{|\mathbb{K}|}}) & \cdots & \mathbb{1}(t_{N_{|\mathbb{K}|}} > t_{N_1})\frac{\partial \Lambda_K^*}{\partial t_{N_1}}(t_{N_{|\mathbb{K}|}}) & \cdots & \frac{\partial \Lambda_K^*}{\partial t_{1_{|\mathbb{K}|}}}(t_{N_{|\mathbb{K}|}}) & \cdots & \lambda_K^*(t_{N_{|\mathbb{K}|}})
\end{bmatrix}
\end{aligned}
$$

2 MODEL DETAILS

Subsection 3.3 introduces the triangular maps we propose for modelling multivariate TPPs. This section provides additional details about their architecture. We also share our source code [1] for those interested in studying our implementation.

All diagrams in this section represent a multivariate TPP with three marks. We represent the input $H_{t_N^-} = \{(t_n, k_n) | 0 \leq t_n < t_N\}$ as a sequence of colored boxes, and, $\{(t_{N,k_1}, k_1), \cdots, (t_{N,|\mathcal{K}|}, k_{|\mathcal{K}|})\}$ as a perpendicular vector at the bottom of the sequence. Given the input, the triangular maps return the transformed data $\tilde{H}_{t_N^-}$ and $\{(\tilde{t}_{N,k_1}, k_1), \cdots, (\tilde{t}_{N,|\mathcal{K}|}, k_{|\mathcal{K}|})\}$. All the maps are invertible. Bear in mind that as the original sequence changes along the chain of triangular maps, the transformed sequence $\tilde{H}_{t_N^-}$ will not necessarily be sorted, and $\tilde{t}_{N,k}$ will differ for different marks.

Pairwise difference ($D : \mathbb{R}_+ \to \mathbb{R}_+$), Figure 1. This operation applies pairwise difference between consecutive timestamps of the same mark. The diagram shows the history split into mark-specific sub-histories. We prepend $t_0 = 0$ and append $t_{N,k}$ to each sub-history to obtain mark-specific pairwise differences. The output is reassembled to maintain the same order as in the input.

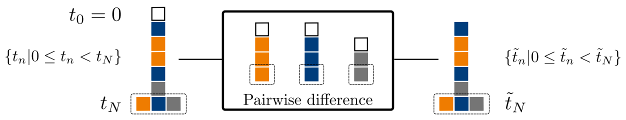
$t_0 = 0$
$\{t_n | 0 \leq t_n < t_N\}$
t_N
Pairwise difference
$\{\tilde{t}_n | 0 \leq \tilde{t}_n < \tilde{t}_N\}$
$\tilde{t}_N$

Figure 1: Pairwise difference

Spline ($\Phi : (0,1) \to (0,1)$), Figure 2. As shown in the diagram, this operation defines a set of mark-specific parameters for the rational-quadratic spline described in Durkan et al. (2019). Splines are flexible non-linear functions $\phi : (0,1) \to (0,1)$ for which the number of parameters depends on the number of spline knots. Like Shchur et al. (2020b), we stack element-wise helper functions before and after splines to ensure domain compatibility. When the domain is $\mathbb{R}_+$ we apply $\psi(x) = 1 - \exp(-x)$ before the spline, and $\psi^{-1}(y) = -\log(1-y)$ after the spline since $\psi : \mathbb{R}_+ \to (0,1)$. When the domain is $\mathbb{R}$ we apply $\sigma(x) = 1/(1+\exp(-x))$ before the spline, and $\sigma^{-1}(y) = \log(y) - \log(1-y)$ after the spline since $\sigma : \mathbb{R} \to (0,1)$. The only occasion in which the domain becomes non-positive is after the use of block-diagonal maps. See Section C.4 of Shchur et al. (2020b) for more details.

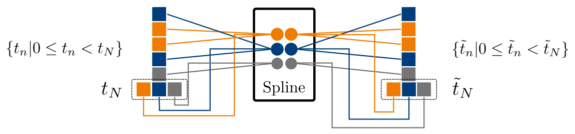
$\{t_n | 0 \leq t_n < t_N\}$
t_N
Spline
$\{\tilde{t}_n | 0 \leq \tilde{t}_n < \tilde{t}_N\}$
$\tilde{t}_N$

Figure 2: Spline

Block-diagonal ($B : \mathbb{R} \to \mathbb{R}$), Figure 3. Shchur et al. (2020b) uses a single block-diagonal matrix for univariate TPPs. Alternatively, we propose the block-diagonal map for multivariate TPPs as a sequence of mark-specific block-diagonal matrices. A block-diagonal matrix contains stacked attention blocks. It outputs a linear combination of the current event with events coming prior to it up to a certain lag, which is defined by the block size and the event position. In that sense, it resembles an attention layer. In contrast to attention, block-diagonal matrices are fixed, but invertible. Block-diagonal maps chained sequentially will have attention blocks shifted by an offset to ensure all events in the sequence can attend to prior events.

Spline transformer ($R : (0,1) \to (0,1)$), Figure 4. This layer is similar to the spline layer above, but instead of having the spline parameters defined internally, spline transformer obtains the spline parameters from the encoded history. A transformer-based network — borrowed from Lin et al. (2022) — encodes the history. The network outputs a vector with the same length as the history of events. We expand each vector row with a linear layer to obtain mark-specific spline parameters. Therefore, each input element will have a tailored parameter

[1] `https://github.com/NUS-IDS/multi-ttpp`

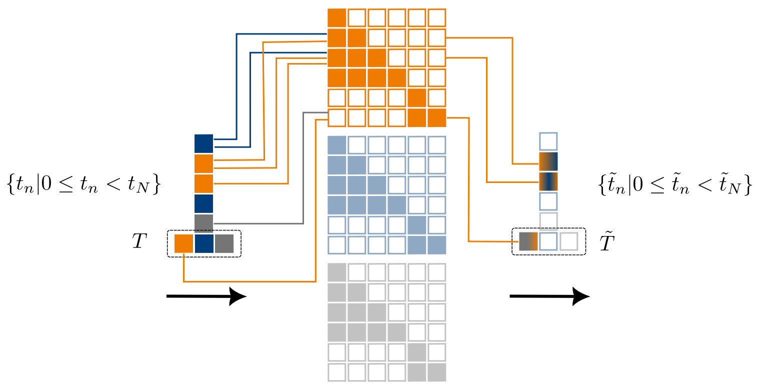
$\{t_n | 0 \leq t_n < t_N\}$
T
$\{\tilde{t}_n | 0 \leq \tilde{t}_n < \tilde{t}_N\}$
$\tilde{T}$

Figure 3: Block-diagonal

set for the spline. This architecture is similar to the RNN architecture proposed by Shchur et al. (2020b) for univariate TPPs.

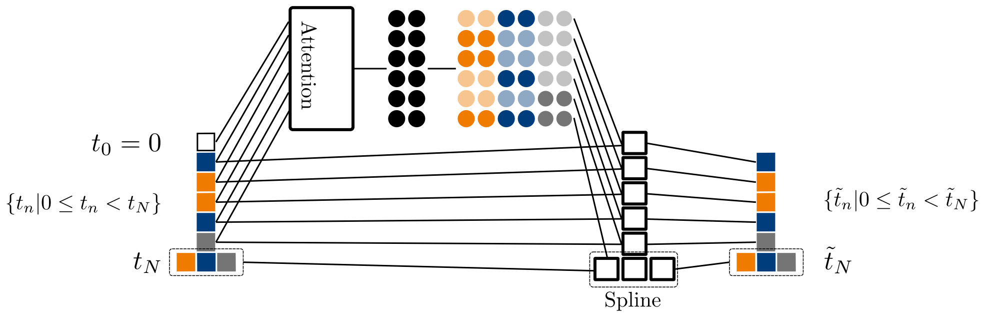
Attention
$t_0 = 0$
$\{t_n | 0 \leq t_n < t_N\}$
t_N
$\{\tilde{t}_n | 0 \leq \tilde{t}_n < \tilde{t}_N\}$
$\tilde{t}_N$
Spline

Figure 4: Spline transformer

3 DATA

This section provides additional details for the datasets used in our benchmarks. Additionally, see the descriptive statistics in Table 1.

		Yelp	***Hawkes***	***Retweet***	***SO***	***MIMIC***	***MOOC***
K		1	2	3	22	75	97
	Train	200	3,200	20,000	4,633	527	4,510
B	**Val.**	40	400	2,000	700	58	1,127
	Test	60	400	2,000	1,300	65	1,410
	Mean	580	132	109	72	4	56
N	**Min.**	380	68	50	41	2	4
	Max.	835	269	264	736	33	493

Table 1: Descriptive statistics. K refers to the total number of marks, B to the total number of sequences, and N to the number of events in a sequence.

Yelp. Sequences of restaurant check-ins from Yelp. When a user visits the premises of a business listed on Yelp, they can check in with them via the mobile app. The data only contains the timestamp of check-ins. It is used in Lin et al. (2022) and available from their source repo [2].

[2]`https://github.com/BIRD-TAO/GNTPP`

Hawkes. Sequences of simulations of a two-dimensional Hawkes process with exotic kernel. The dataset was first introduced in Zhang et al. (2020). They generate it with Python library `tick` [3]. The authors also provide the equations for the kernels used in the simulation. We obtain the data from their source repo [4]. The data is also benchmarked in Zhang et al. (2021).

Retweet. A sample collected from 7 October to 7 November 2011 of retweet sequences, each retweet event contains the tweet timestamp and the popularity of the user who tweeted it. Users belong to one of three categories determined by their number of followers. The dataset was first introduced by Zhao et al. (2015). Later, it was prepared for benchmarking neural TPPs by Mei and Eisner (2017) — see Section C.5 of their Supplementary Material for additional information. This dataset was also benchmarked in Zuo et al. (2020) and Zhang et al. (2020) — from whose source repo [5] we obtain the dataset — Zhang et al. (2021) and Lin et al. (2022).

SO. Sequences of StackOverflow user awards received between 1 January 2012 and 1 January 2014. Users receive awards after completing certain tasks on the question-answering website that names the dataset. The dataset was first introduced by Bacchelli (2013). Later, it was prepared for benchmarking neural TPPs by Du et al. (2016), see Section 6.3. This dataset was also benchmarked in Mei and Eisner (2017), Zuo et al. (2020), Shchur et al. (2020a), Zhang et al. (2020), Zhang et al. (2021), and Lin et al. (2022) —- from whose source repo [6] we obtain the dataset.

MIMIC. Sequences of anonymous patient's clinical visits in seven years, each event contains its timestamp and one out of 75 possible clinical diagnostics. The dataset is extracted from the Multiparameter Intelligent Monitoring in Intensive Care II database (Lee et al., 2011). This dataset was benchmarked in Du et al. (2016), Mei and Eisner (2017), Zuo et al. (2020), Zhang et al. (2020) — from whose source repo [7] we obtain the dataset — and Zhang et al. (2021).

MOOC. Sequences of interaction with an online course system, each event contains its timestamp and one out of 97 possible interactions. The dataset, extracted from the XuetangX MOOC platform, debuted in the KDD Cup 2015. It contains data from 40 different courses. The data is described in more detail by Kuzilek et al. (2017) and Kumar et al. (2019). It was first used for benchmarking neural TPPs by Shchur et al. (2020a). This dataset was also benchmarked in Lin et al. (2022) — from whose source repo [8] we obtain the dataset.

4 BENCHMARK METRICS

We benchmark three different metrics, which are listed below. In all cases, assume that the test set contains a total of B sequences such that the total number of events in the set is $\sum_{b=1}^{B} N_b$.

Quantile-quantile deviation (*QQD*). We compute the average absolute difference between the estimated and expected percentiles of the compensated inter-event duration. The percentiles of the exponential distribution with unit-rate for $p = F(t) \in (0.01, \cdots, 0.99)$ are computed as follows.

$$Q_p \equiv F^{-1}(t) = -\log(1 - F(p)) \tag{2}$$

Let event (t_n, k_n) such that the n-th event corresponds to the m_{k_n}-th k-marked event and $t_0 = 0$, the compensated inter-event duration is given below.

$$\Delta\tilde{t}_n = \Delta\tilde{t}_{m_{k_n}} = \Lambda_k^*\left(t_{m_{k_n}}\right) - \Lambda_k^*\left(t_{(m-1)_{k_n}}\right) \tag{3}$$

To compute the empirical percentiles, we append all compensated inter-event durations from all sequences in the test set to a single vector $\Delta\tilde{t}$ which we sort. Given the cumulative distribution p and $h = (\sum_{b=1}^{B} N_b + 1)p$, the empirical quantile is equal to:

$$\hat{Q}_p = \Delta\tilde{t}_{\lfloor h \rfloor} + (h - \lfloor h \rfloor) \times (\Delta\tilde{t}_{\lfloor h \rfloor} - \Delta\tilde{t}_{\lfloor h-1 \rfloor}) \tag{4}$$

[3] `https://x-datainitiative.github.io/tick`
[4] `https://github.com/QiangAIResearcher/sahp_repo`
[5] `https://github.com/QiangAIResearcher/sahp_repo`
[6] `https://github.com/BIRD-TAO/GNTPP`
[7] `https://github.com/QiangAIResearcher/sahp_repo`
[8] `https://github.com/BIRD-TAO/GNTPP`

The mean absolute quantile is computed as:

$$\text{QQD} = \frac{1}{99} \sum_{p \in (0.01, \cdots, 0.99)} |Q_p - \hat{Q}_p| \tag{5}$$

We train the benchmark models listed in Section 5 of the Supplementary Materials using Lin et al. (2022) implementation [9]. However, their implementation of *QQD* is incorrect. We can split the models we benchmark against into two groups. The first group implements the compensator in closed-form — *RMTPP*, *LogNorm*, *ERTPP*, *FNNInt*, and *WeibMix* — or with Monte Carlo approximation — *SAHP*, *THP*. Models in this group return the compensated inter-event interval as in Equation 3.

The second group does not implement the compensator — *DETER*, *TCDDM*, *TCVAE*, *TCGAN*, *TCCNF*, *TCNSN*. The compensated inter-event duration must be approximated via sampling. To obtain these values, models in this group should first compute an empirical approximation to the cumulative distribution function of the compensated inter-event duration. However, the implementation [10] is incorrect. Rather than the cumulative distribution function, their implementation attempts to approximate the probability density function as follows. It evenly splits the domain of inter-arrival times up to an upper bound. Inter-arrival time are sampled and the fraction of samples that fall in the interval containing the true inter-arrival time is computed. This routine cannot approximate the probability density function of the compensated inter-arrival time because the regularity of the splits in the original domain is not preserved after transformation. Intervals that cover peaks of the conditional intensity function will be expanded in the compensated domain and vice-versa. The alternative is to approximate the cumulative distribution function as follows. We sample inter-arrival times. The approximated cumulative distribution function is the fraction of samples equals to or smaller than the true value. Using the cumulative distribution function of the exponential distribution with unit rate, we can invert the approximated cumulative distribution to obtain the compensated inter-event duration.

Other issues remain. In both cases, when calling the method to compute the compensator, the implementation passes the last inter-event interval contained in the history of events instead of the next event inter-event interval [11]. Finally, they use the same routine to compute the quantiles [12], even though they pass values from different domains depending on whether models are members of the first or second group above. Regardless, this routine fails to compute *QQD* whether we pass the compensated inter-arrival times or the cumulative distribution, which one can easily verify by passing values sampled from an exponential distribution with unit rate. We fix all these issues in our fork of their repo [13] to obtain consistent *QQD* metrics.

Mean absolute percentage error (*MAPE*). Let the predicted inter-event duration $\Delta\hat{t}_{n,b}$, the true duration $\Delta t_{n,b}$ and $\Delta\bar{t}$ be the maximum duration observed during training, the *MAPE* is defined below.

$$\text{MAPE} = \frac{1}{\sum_{n=1}^{B} N_b} \sum_{b=1}^{B} \sum_{n=1}^{N_b} \frac{|\max(\Delta\hat{t}_{n,b}, \Delta\bar{t}) - \Delta t_{n,b}|}{\Delta t_{n,b} + \epsilon} \tag{6}$$

We add $\epsilon \equiv 1e-7$ to the denominator to ensure the expression is well-defined even for $\Delta t_{n,b} = 0$ as it occurs on the Twitter dataset on rare occasions. The magnitude of ϵ can significantly affect the *MAPE*, so we use the same as in Lin et al. (2022).

All the benchmarked models — except for *DETER* — are probabilistic, which rather than returning a single prediction, returns a distribution of possible inter-event durations. Lin et al. (2022) define the predicted inter-event duration as the mean inter-event duration of the model. They clamp the mean inter-event duration to equal to or less than the maximum inter-event duration observed during training. All predicted values are computed on the scaled data in which the original timestamps are divided by the maximum timestamp observed

[9] `https://github.com/BIRD-TAO/GNTPP`

[10] `https://github.com/BIRD-TAO/GNTPP/blob/2042c9be947d0e600ceb98a1eb55cfe3d55be33c/models/prob_decoders/base_prob_dec.py#L58`

[11] `https://github.com/BIRD-TAO/GNTPP/blob/2042c9be947d0e600ceb98a1eb55cfe3d55be33c/models/tpp_warper.py#L191`

[12] `https://github.com/BIRD-TAO/GNTPP/blob/2042c9be947d0e600ceb98a1eb55cfe3d55be33c/trainers/trainer.py#L225`

[13] Our fork is available at `https://github.com/gzagatti/GNTPP`.

during training and multiplied by the normalization scale which is set to 50. We follow the same conventions for consistency's sake.

There are three alternatives to compute the mean. The first alternative is to derive it analytically. That is the case for *LogNorm*, *ERTPP*, and *WeibMix*. The second alternative consists of computing the mean via Riemann sum. Lin et al. (2022) approximate the distribution of inter-event duration with a discrete partition of the original time domain up to an upper bound. The mean is computed as a weighted sum. The upper bound ensures the mean is constrained to 90 percent of the maximum training timestamp. While this constraint is very generous for the inter-event duration, it limits extreme values at the tail of the distribution. *RMTPP*, *SAHP*, and *THP* compute the mean with this approach. The third alternative consists of estimating the mean via sampling. Given S samples, the predicted inter-event duration is equal to the average sampled duration.

$$\Delta\hat{t}_{n,b} = \frac{1}{S}\sum_{s\in S}\Delta\hat{t}_{n,b,s} \tag{7}$$

That is the case for *TCCDM*, *TCVAE*, *TCGAN*, *TCCNF*, *TCNSN*, and the methods we present in this paper. While our methods can approximate the expected inter-event duration via the Riemann sum similar to the methods above, we predict the mean via sampling. As discussed above in the context of *QQD*, the even partition of the original time domain is not necessarily even in the compensated domain, which can bias the mean when approximating it via the Riemann sum. For instance, in the *Retweet* dataset, most of the events happen right at the start, while there is a very long tail for the last few events — see Figures 8 below. Using the Riemann sum would distort the mean approximation by placing too much weight on rare events at the tail.

Accuracy of top-1 (*TOP1*) and top-3 (*TOP3*) mark prediction. Given a ranking of mark predictions $\hat{k}_{n,b} = \{\hat{k}_{(1),n,b}, \cdots, \hat{k}_{(R),n,b}\}$, the top-$R$ accuracy is the fraction of time that the true mark appears within the top-R predictions.

$$\text{TOPR} = \frac{1}{\sum_{n=1}^{B} N_b}\sum_{b=1}^{B}\sum_{n=1}^{N_b}\mathbb{1}\left(k_{n,b}\in\hat{k}_{n,b}\right) \tag{8}$$

As discussed in Section 3.1, the probability density function of marks is equal to $f^*_K(k \mid t) = \lambda^*_k(t)/\lambda^*_g(t)$. Given the predicted event time $\hat{t}_n = t_{n-1} + \Delta\hat{t}_n$, we define the ranking of mark predictions $\hat{k}_{n,b} = \{\hat{k}_{(1),n,b}, \cdots, \hat{k}_{(R),n,b}\}$ such that $\lambda^*_{\hat{k}_{(r),n,b}}(\hat{t}_{n,b}) \geq \lambda^*_{\hat{k}_{(q),n,b}}(\hat{t}_{n,b})$ for $r < q$. Alternatively, Zhang et al. (2020) defines the r-th mark prediction as mark k which obtains the r-th highest $\Lambda^*_k(\bar{T})$ given an upper-bound $\bar{T}$. Our proposed definition suggests an approach in which the user first predicts the next-event time, and then selects the most likely mark.

Lin et al. (2022) assumes that mark distribution is independent of t given the learned history representation. As we discussed in Section 3.1, this assumption implies a constant likelihood between marks, that is, $f^*_K(k \mid t) = f^*_K(k)$. They define the r-th mark prediction as mark k which obtains the r-th highest $f^*_K(k)$. As discussed in Supplementary Materials Section 2, Lin et al. (2022) has a separate model for the mark distribution, which is independent of the conditional intensity.

5 BENCHMARK MODELS

We benchmark the models proposed in our paper against the models tested in Lin et al. (2022), which include most of the state-of-the-art neural TPP models proposed in the literature. The authors make their source code freely available [14]. The advantage of using the same repo is that all models use the same trainer and utilities — from loading the data to computing metrics —, which ensures that model predictions are comparable. We also implement our proposed models to benefit from these shared facilities.

For convenience, we list the models that we benchmark against. In all cases, Lin et al. (2022) uses an attention encoder to encode the history of events. The decoders model the univariate, ground TPP and follow the original sources. This is what is listed below. For the mark distribution, Lin et al. (2022) uses the same approach for all models. They assume that the mark distribution is independent of the time distribution. The mark distribution follows from the logit of the marks, which is computed from the embedded history using a linear projection and the softmax. All their models are trained with the objective of minimizing the sum of the negative log-likelihood of the ground TPP plus the mark loss, which equals the cross-entropy loss.

[14] `https://github.com/BIRD-TAO/GNTPP`

Lin et al. (2022) implementation of *SAHP* and *THP* differ from the original ones in the computation of the mark distribution [15], since the original versions of these models produce a multivariate conditional intensity. Despite these changes, their versions of *SAHP* and *THP* remain competitive with respect to mark accuracy. Therefore, the changes are not necessarily detrimental to performance.

Deterministic (*DETER*). A baseline deterministic decoder uses a linear projection whose weights are all constrained to be positive. The ground TPP is trained as a regression problem to minimize the mean-squared error between the true and predicted timestamp.

Recurrent marked TPP (*RMTPP*) (Du et al., 2016). This decoder pioneered neural TTPs. It models the conditional intensity function λ_g^* according to a Gompertz distribution as discussed by Lin et al. (2021). Lin et al. (2022) extends by modelling λ_g^* as a mixture of Gompertz distributions. The compensator Λ_g^* can be analytically computed.

Log-Normal mixture (*LogNorm*) (Shchur et al., 2020a). This decoder was proposed by Shchur et al. (2020a) in the context of "intensity-free" neural TPP models — that is, models defined by the inter-event duration conditional distribution. It is obviously the case that λ_g^* is defined for any conditional distribution, but the authors choose instead to focus on the conditional distribution for simplicity purposes. *LogNorm* models λ_g^* as a mixture of Log-Normal distributions. Λ_g^* can be analytically computed. Although Λ_g^* can be analytically computed, there is no stable implementation of the survival function of the Gaussian distribution. For computational purposes, Lin et al. (2022) clamps the Gaussian cumulative density function at both extremes.

Event recurrent TPP (*ERTPP*) (Xiao et al., 2017). Xiao et al. (2017) adopt a Gaussian penalty function to compute the loss of the predicted timestamp. As discussed in Lin et al. (2021), the approach is equivalent to modelling λ_g^* as a Gaussian distribution, which they extend to a mixture of Gaussian distributions. Although Λ_g^* can be analytically computed, there is no stable implementation of the survival function of the Gaussian distribution. For computational purposes, Lin et al. (2022) clamps the Gaussian cumulative density function at both extremes.

Fully neural network intensity (*FNNInt*) (Omi et al., 2019). In this case, the decoder describes the compensator Λ_g^*. Constraints are in-place to ensure the network satisfies conditions on Λ_g^* such as non-negativity and strict monotonicity. The conditional intensity comes from the automatic differentiation of Λ_g^*. Originally, the model only applied to univariate TPPs. Lin et al. (2021) extends this method to marked TPPs.

Weibull mixture (*WeibMix*) (Lin et al., 2021). The decoder is a mixture of Weibull distributions. The Weibull distribution commonly features in survival analysis as discussed in Daley and Vere-Jones (2003) Chapter 1. The compensator Λ_g^* can be analytically computed.

Self-attentive Hawkes process (*SAHP*) (Zhang et al., 2020). This decoder takes inspiration from the Hawkes process to model the conditional intensity, $\lambda_k^*(t) = \text{softplus}\left(\alpha_k^* + (\beta_k^* - \alpha_k^*)\exp(-\gamma_k^* \times (t - t_{n-1}))\right)$. For each mark k, the parameter $\theta_k^* = (\alpha_k^*, \beta_k^*, \gamma_k^*)$ is a neural network that takes the encoded history as input and outputs a vector of dimension three. Therefore, *SAHP* models marked TPPs as a multivariate TPP. Λ_k^* must be approximated via Monte Carlo integration.

Transformer Hawkes process (*THP*) (Zuo et al., 2020). This decoder takes inspiration from the Hawkes process to model the conditional intensity, $\lambda_k^*(t) = \text{softplus}\left(\alpha_k^* + \beta_k^* \times ((t - t_{n-1})/t_{n-1})\right)$. For each mark k, the parameter $\theta_k^* = (\alpha_k^*, \beta_k^*)$ is a neural network that takes the encoded history as input and outputs a vector of dimension two. Therefore, *THP* models marked TPPs as a multivariate TPP. Λ_k^* must be approximated via Monte Carlo integration.

Temporal conditional diffusion denoising model (*TCDDM*) (Lin et al., 2022). This decoder models the conditional distribution as a diffusion model. It assumes that inter-event duration is distributed according to a sequence of latent variables, whose posterior distribution follows a sequence of Normal distributions. The seed distribution at the end of the sequence follows a Gaussian distribution with zero mean and unit standard deviation. Given a sample from the seed distribution, we obtain the distribution of the original inter-event duration as a sequence of Gaussian distribution whose parameters are the output of a neural network — the denoising network. A variational bound approximates the log-likelihood of the data. With the reparametrization trick, it is possible to estimate the variational bound by sampling the seed distribution. Λ_g^* is not implemented,

[15]The time distribution is not affected.

though it could be approximated with the posterior.

Temporal conditional variational autoencoder (*TCVAE*) (Lin et al., 2022). This decoder models the conditional distribution with a variational autoencoder. It uses an encoder network to approximate the inter-event duration posterior distribution — that is, to approximate the mean and standard deviations. The posterior can be interpreted as an approximation to the prior Gaussian distribution with zero mean and unit standard deviation. It uses the posterior to sample in the transformed space. Conversely, samples from the transformed space are transformed back to the original space with a decoder network. The evidence lower bound approximates the log-likelihood of the data. Λ_g^* is not implemented, though it could be approximated with the encoder network and the posterior distribution.

Temporal conditional generative adversarial network (*TCGAN*) (Lin et al., 2022). This decoder models the conditional distribution much like the decoder network of *TCVAE*. However, this transformation is not invertible because no encoding network needs to be defined. The model is trained via an adversarial process in which the discriminator maximizes the Wasserstein distance between the distributions of generated inter-event intervals and of true inter-event intervals. The authors add a Lipschitz constraint as a regularization term. Λ_g^* is not implemented.

Temporal conditional continuous normalizing flows (*TCCNF*) (Lin et al., 2022). The decoder transforms draws from a Gaussian distribution $\epsilon \sim \text{Normal}(0, 1)$ using a neural ordinary differential equation (ODE), that is $\Delta t_n = \epsilon + \int_0^L G^*(l \mid \theta)\, dl$, such that $G^*(l \mid \theta)$ is a neural network, which can be interpreted as the derivative of the transformation with respect to layer depth. The log-likelihood of the data derives from the change of random variable formula for probability densities. This method shares many similarities with the methods we proposed in this paper. While we define our models as a discrete sequence of triangular maps, the neural ODE is equivalent to a continuous sequence. However, *TCCNF* fails to use the time-change theorem as we propose. First, it uses a Gaussian distribution instead of the exponential as the base distribution — though this might not make much difference given that there is a one-to-one mapping between these distributions. Second, *TCCNF* is trained by sampling ϵ and transforming it back to the original space. The true timestamp thus only affects the computation of the Jacobian. In our proposed approach, the true timestamp is first transformed into a sample of the exponential distribution. The timestamp then affects all the elements of the log-likelihood, which affects training since there is a trade-off between increasing the likelihood of the observed points and the likelihood of not observing the points in between. Λ_g^* is not implemented, though it could be computed by reversing the neural ODE and under the assumption that G^* is strictly monotonic.

Temporal conditional noise score network (*TCNSN*) (Lin et al., 2022). The decoder is modelled via score matching, which learns the gradient field — or scores — of the target distribution rather than the distribution itself. The score is defined as a neural network. The objective function consists of minimizing the score function with respect to the predicted timestamp plus the mean-squared error between the true and predicted timestamp. Λ_g^* is not implemented.

6 ADDITIONAL BENCHMARKS

	Number of Parameters						Loss per event (↓)					
	Yelp	*Hawkes*	*Retweet*	*SO*	*MIMIC*	*MOOC*	*Yelp*	*Hawkes*	*Retweet*	*SO*	*MIMIC*	*MOOC*
MultiMRP	**63**	**605**	**187**	**1,365**	22,651	29,295	13.73	1.83	6.49	15.70	1.11	14.00
MultiTriTPP	<u>998</u>	<u>1,275</u>	<u>7,696</u>	48,511	34,126	222,616	13.42	1.60	6.09	15.41	-0.60	14.06
MultiTraTPP	2,432	15,038	11,728	26,478	438,904	643,878	13.52	1.41	5.63	15.13	0.41	12.27
DETER	13,647	13,696	13,745	<u>14,676</u>	**17,273**	**18,351**	0.01	0.88	2.22	2.20	40.45	3.05
RMTPP	15,294	15,343	15,392	16,323	18,920	19,998	-1.51	1.08	-1.94	4.98	34.86	1.92
LogNorm	15,294	15,343	15,392	16,323	18,920	19,998	-1.79	1.08	-2.65	4.95	5.78	1.40
ERTPP	15,294	15,343	15,392	16,323	18,920	19,998	-1.42	1.10	-1.01	5.02	37.74	3.50
WeibMix	15,294	15,343	15,392	16,323	18,920	19,998	-1.79	0.56	-2.61	3.88	3.19	1.23
FNNInt	14,735	14,784	14,833	15,764	<u>18,361</u>	<u>19,439</u>	-0.95	0.47	-2.66	2.00	3.76	-0.52
SAHP	13,713	13,861	14,009	16,821	24,665	27,921	-1.65	0.40	-3.00	1.89	3.20	-2.19
THP	13,647	13,729	13,811	15,369	19,715	21,519	-1.61	0.43	-1.30	1.95	2.82	0.12
TCDDM	18,911	18,960	19,009	19,940	22,537	23,615	0.25	1.23	1.76	2.28	1.47	4.03
TCVAE	21,103	21,152	21,201	22,132	24,729	25,807	2.30	1.83	2.02	2.55	1.69	3.65
TCGAN	15,823	15,872	15,921	16,852	19,449	20,527	0.04	0.10	0.05	0.05	0.04	0.01
TCCNF	17,075	17,124	17,173	18,104	20,701	21,779	-1.63	0.45	-2.83	2.09	2.70	24.29
TCNSN	15,775	15,824	15,873	16,804	19,401	20,479	0.04	0.72	0.80	1.43	0.67	2.11

Table 2: Number of trainable parameters and loss per event on test set. Best value is bold and second-best is underlined. Losses are not necessarily comparable across every model.

	TOP1 (↑)						**TOP3** (↑)					
	Yelp	***Hawkes***	***Retweet***	***SO***	***MIMIC***	***MOOC***	***Yelp***	***Hawkes***	***Retweet***	***SO***	***MIMIC***	***MOOC***
Multi-TraTPP	1.00	0.53	0.55	0.50	0.26	0.05	1.00	1.00	1.00	0.82	0.48	0.12
Multi-TriTPP	1.00	0.50	0.53	0.33	0.19	0.10	1.00	1.00	1.00	0.55	0.47	0.24
Mul-tiMRP	1.00	0.52	0.49	0.40	0.27	0.09	1.00	1.00	1.00	0.69	0.44	0.21
DETER	1.00	0.55	0.57	**0.53**	**0.59**	0.33	1.00	1.00	1.00	**0.84**	0.61	0.66
RMTPP	1.00	0.55	0.60	**0.53**	0.57	0.33	1.00	1.00	1.00	**0.84**	0.61	0.66
Log-Norm	1.00	0.85	0.60	**0.53**	**0.59**	0.33	1.00	1.00	1.00	**0.84**	**0.62**	0.65
ERTPP	1.00	0.55	0.60	**0.53**	0.58	0.32	1.00	1.00	1.00	**0.84**	0.61	0.65
WeibMix	1.00	0.54	**0.68**	**0.53**	0.57	0.25	1.00	1.00	1.00	**0.84**	0.59	0.49
FNNInt	1.00	0.91	0.60	**0.53**	0.58	0.33	1.00	1.00	1.00	**0.84**	0.61	0.66
SAHP	1.00	**0.95**	0.60	**0.53**	**0.59**	0.32	1.00	1.00	1.00	**0.84**	**0.62**	0.65
THP	1.00	**0.95**	0.59	0.52	**0.59**	0.33	1.00	1.00	1.00	**0.84**	**0.62**	0.65
TCDDM	1.00	0.55	0.60	**0.53**	**0.59**	0.33	1.00	1.00	1.00	**0.84**	**0.62**	0.66
TCVAE	1.00	0.57	N/A	**0.53**	0.58	0.33	1.00	1.00	1.00	**0.84**	0.61	0.66
TCGAN	1.00	0.52	N/A	**0.53**	0.58	**0.40**	1.00	1.00	1.00	0.83	**0.62**	0.72
TCCNF	1.00	0.57	0.60	**0.53**	**0.59**	0.33	1.00	1.00	1.00	**0.84**	0.61	0.66
TCNSN	1.00	0.55	0.60	**0.53**	**0.59**	0.33	1.00	1.00	1.00	**0.84**	0.61	0.66

Table 3: Test set model evaluation. Top-1 accuracy (*TOP1*) and top-3 accuracy (*TOP3*). Best value is bold and second-best is underlined.

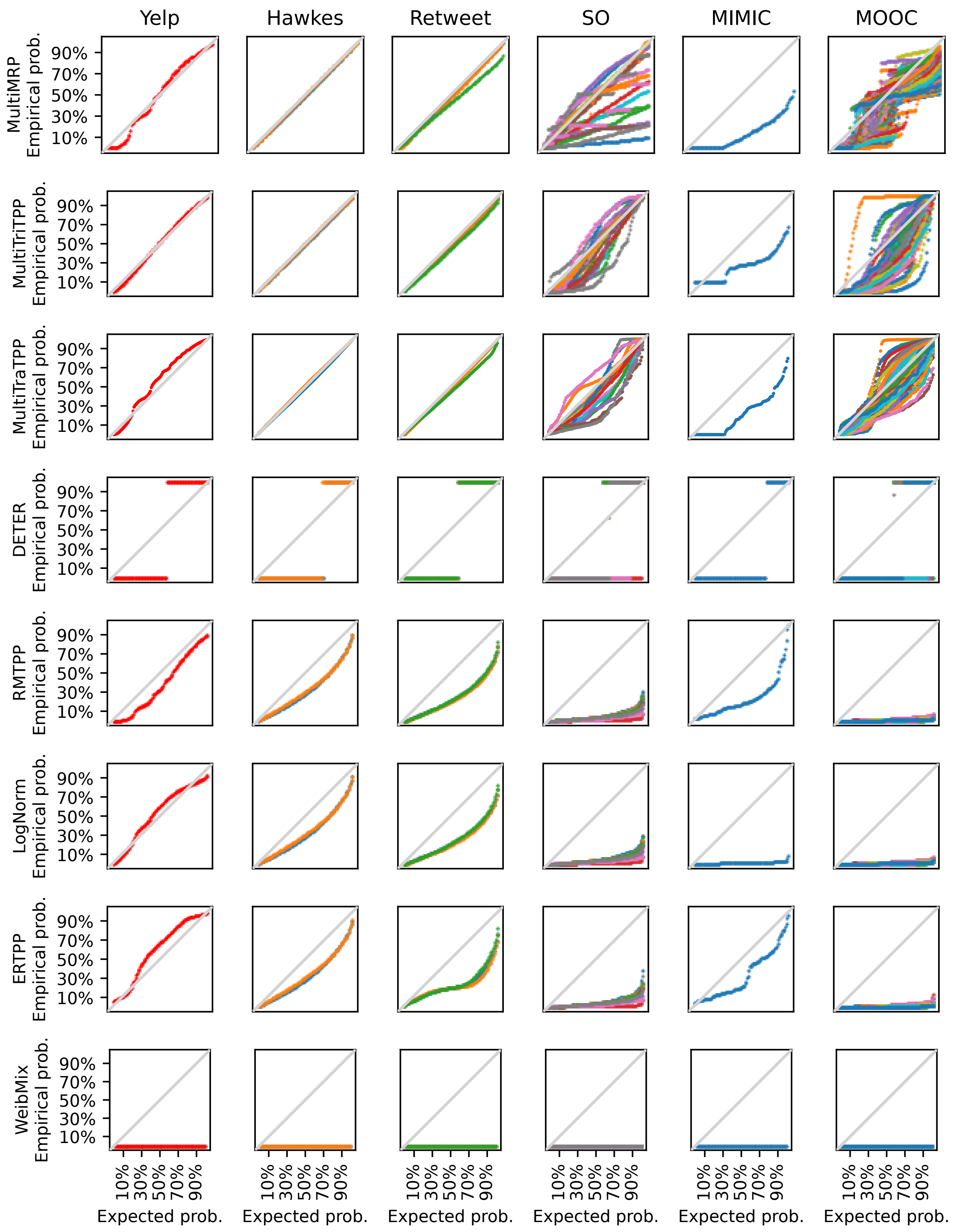

Figure 5: Q-Q plot: expected versus predicted mark-specific quantiles distinguished by colors. We only show quantiles of marks with at least 50 observations in the test set, so not all marks are represented in *SO* (18 out 22 marks), *MIMIC* (1 out of 75 marks), and *MOOC* (81 out of 97 marks).

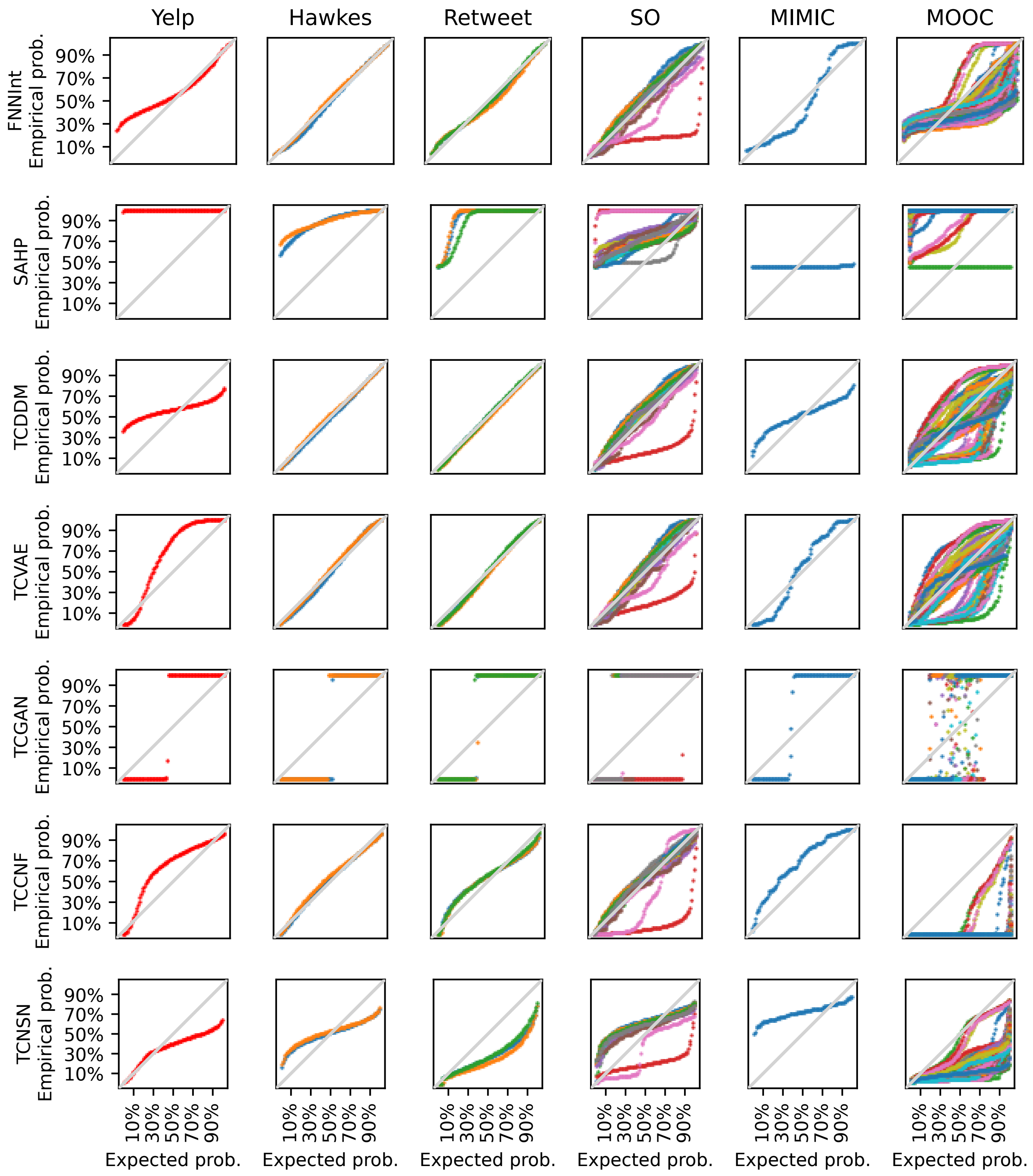

Figure 5: Q-Q plot: expected versus predicted mark-specific quantiles distinguished by colors. We only show quantiles of marks with at least 50 observations in the test set, so not all marks are represented in *SO* (18 out 22 marks), *MIMIC* (1 out of 75 marks), and *MOOC* (81 out of 97 marks).

7 MISCELLANEOUS

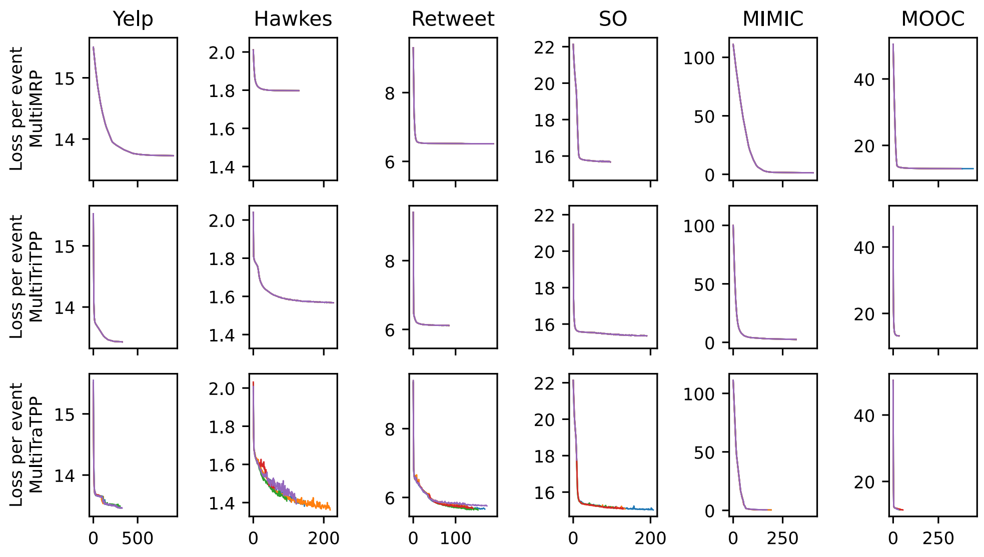

Figure 6: Convergence of validation losses with different initial random seeds. Both *MultiMRP* and *MultiTriTPP* are initialized with non-random parameters which follows Shchur et al. (2020b), so we do not observe any variation in the training routine.

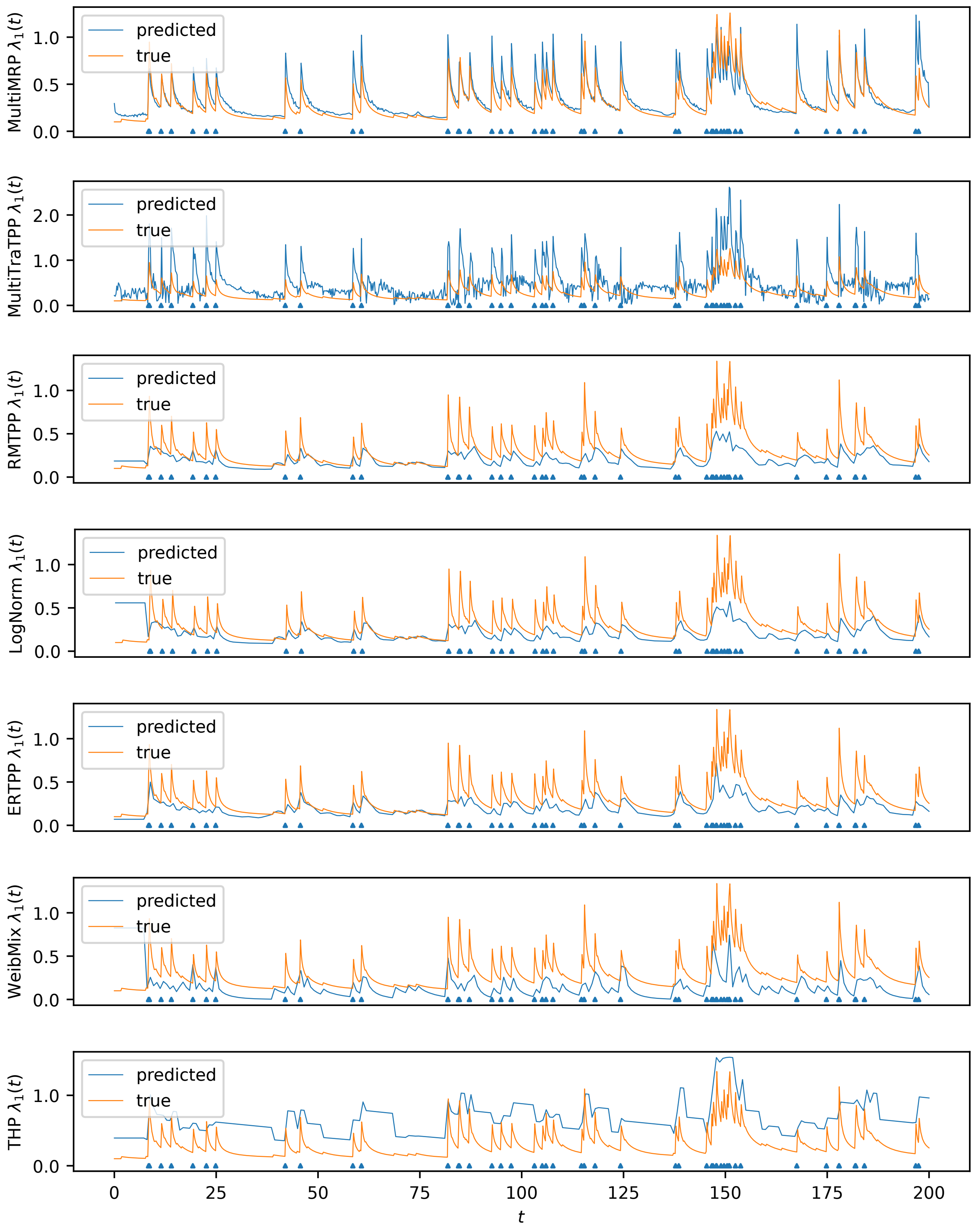

Figure 7: Predicted versus true intensity rate, Hawkes process. Sequence chosen at random from the test set. For illustration purposes, we only depict the evolution of the first mark. Marks at the bottom represent the true event timestamps only for the first mark. Only models whose source code implements the conditional intensity shown.

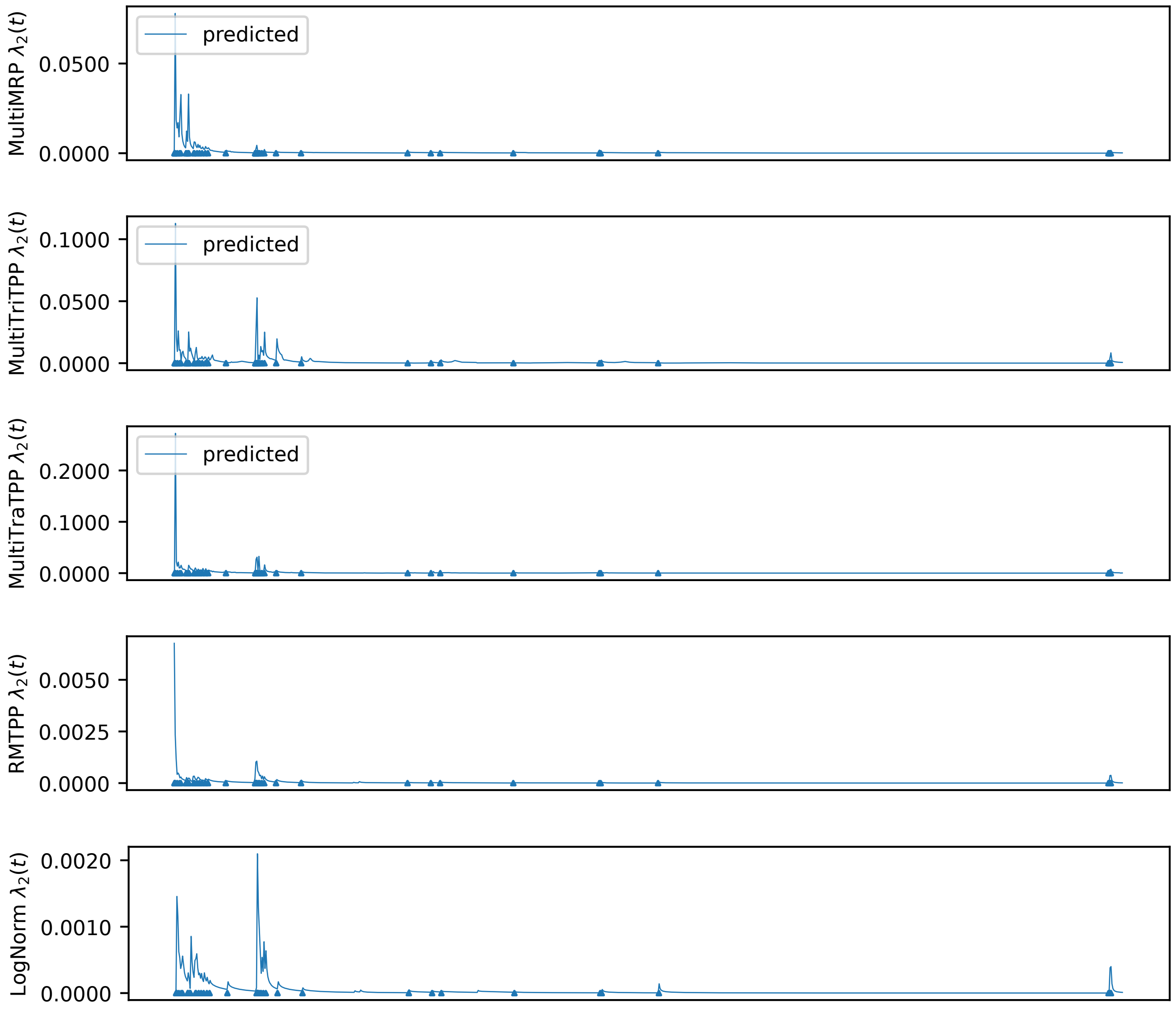

Figure 8: Predicted intensity rate, *retweets*. Sequence chosen at random from the test set. For illustration purposes, we only depict the evolution of the second mark. Marks at the bottom represent the true event timestamps only for the second mark. The plot is truncated at $70,000$, while the true sequence ends at $499,915$. Only models whose source code implements the conditional intensity shown.

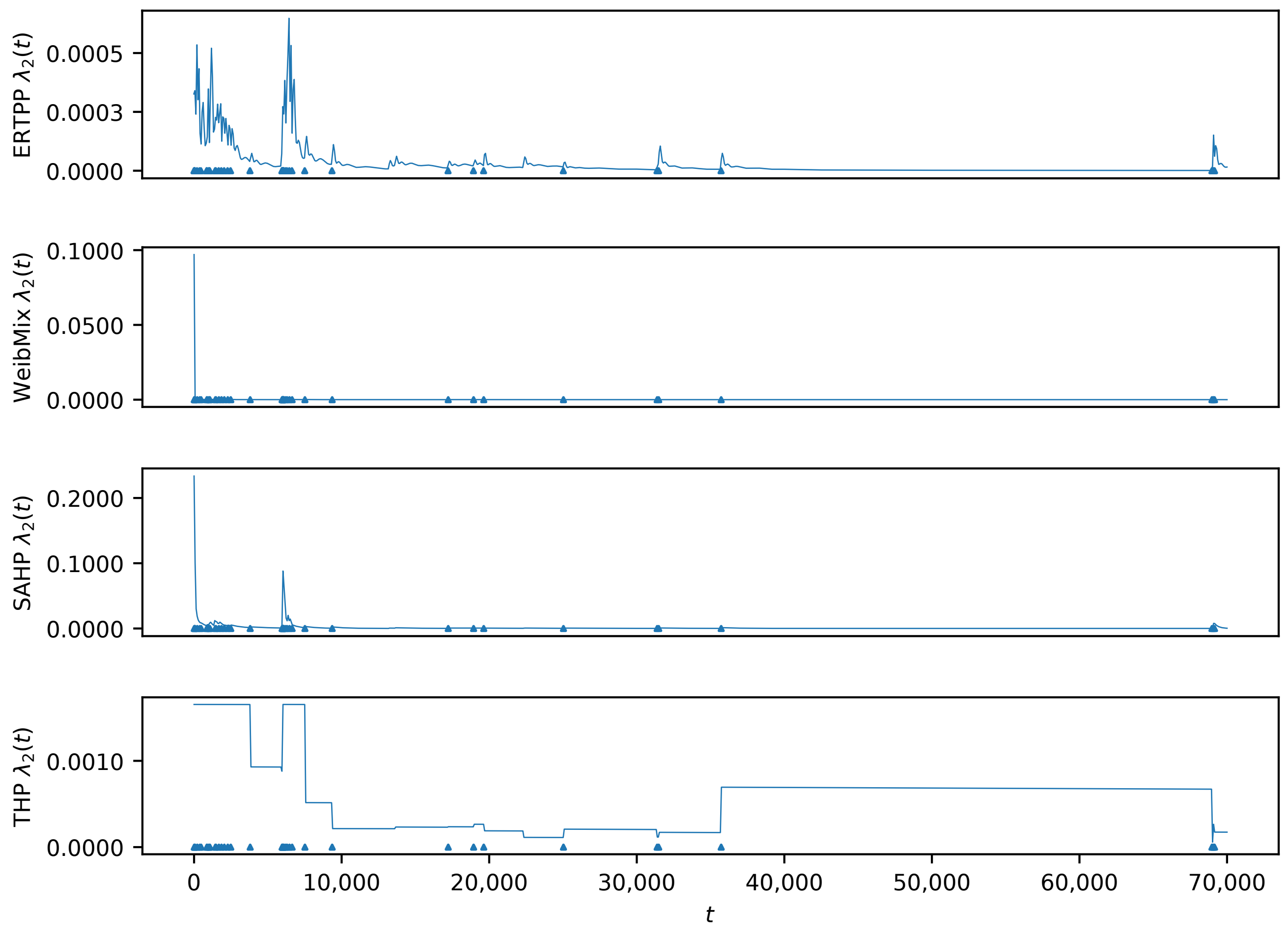

Figure 8: Predicted intensity rate, *retweets*. Sequence chosen at random from the test set. For illustration purposes, we only depict the evolution of the second mark. Marks at the bottom represent the true event timestamps only for the second mark. The plot is truncated at 70, 000, while the true sequence ends at 499, 915. Only models whose source code implements the conditional intensity shown.

References

Bacchelli, A. (2013). Mining challenge 2013: Stack overflow. In *The 10th Working Conference on Mining Software Repositories*.

Daley, D. J. and Vere-Jones, D. (2003). *An Introduction to the Theory of Point Processes: Volume I: Elementary Theory and Methods*. Probability and Its Applications, An Introduction to the Theory of Point Processes. Springer-Verlag, New York, 2 edition.

Du, N., Dai, H., Trivedi, R., Upadhyay, U., Gomez-Rodriguez, M., and Song, L. (2016). Recurrent Marked Temporal Point Processes: Embedding Event History to Vector. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, New York, NY, USA. Association for Computing Machinery.

Durkan, C., Bekasov, A., Murray, I., and Papamakarios, G. (2019). Neural Spline Flows.

Kumar, S., Zhang, X., and Leskovec, J. (2019). Predicting Dynamic Embedding Trajectory in Temporal Interaction Networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, Anchorage AK USA. ACM.

Kuzilek, J., Hlosta, M., and Zdrahal, Z. (2017). Open University Learning Analytics dataset. *Scientific Data*, 4(1).

Lee, J., Scott, D. J., Villarroel, M., Clifford, G. D., Saeed, M., and Mark, R. G. (2011). Open-access MIMIC-II database for intensive care research. In *2011 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*.

Lin, H., Tan, C., Wu, L., Gao, Z., and Li, S. Z. (2021). An Empirical Study: Extensive Deep Temporal Point Process.

Lin, H., Wu, L., Zhao, G., Pai, L., and Li, S. Z. (2022). Exploring Generative Neural Temporal Point Process. *Transactions on Machine Learning Research*.

Mei, H. and Eisner, J. M. (2017). The Neural Hawkes Process: A Neurally Self-Modulating Multivariate Point Process. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Omi, T., Ueda, N., and Aihara, K. (2019). Fully neural network based model for general temporal point processes. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, number 190. Curran Associates Inc., Red Hook, NY, USA.

Shchur, O., Biloš, M., and Günnemann, S. (2020a). Intensity-free learning of temporal point processes. *arXiv:1909.12127 [cs, stat]*.

Shchur, O., Gao, N., Biloš, M., and Günnemann, S. (2020b). Fast and Flexible Temporal Point Processes with Triangular Maps. In *Advances in Neural Information Processing Systems*, volume 33. Curran Associates, Inc.

Xiao, S., Yan, J., Chu, S. M., Yang, X., and Zha, H. (2017). Modeling The Intensity Function Of Point Process Via Recurrent Neural Networks.

Zhang, Q., Lipani, A., Kirnap, O., and Yilmaz, E. (2020). Self-Attentive Hawkes Processes. *arXiv:1907.07561 [cs, stat]*.

Zhang, Q., Lipani, A., and Yilmaz, E. (2021). Learning Neural Point Processes with Latent Graphs. In *Proceedings of the Web Conference 2021*, WWW '21, New York, NY, USA. Association for Computing Machinery.

Zhao, Q., Erdogdu, M. A., He, H. Y., Rajaraman, A., and Leskovec, J. (2015). SEISMIC: A Self-Exciting Point Process Model for Predicting Tweet Popularity. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, New York, NY, USA. Association for Computing Machinery.

Zuo, S., Jiang, H., Li, Z., Zhao, T., and Zha, H. (2020). Transformer Hawkes Process. In *Proceedings of the 37th International Conference on Machine Learning*. PMLR.