Preventing Conflicting Gradients in Neural Temporal Point Process Models for Irregular Time Series Data

Tanguy Bosser
Department of Computer Science
University of Mons
tanguy.bosser@umons.ac.be

Souhaib Ben Taieb
Department of Statistics and Data Science
Mohamed bin Zayed University of Artificial Intelligence
souhaib.bentaieb@mbzuai.ac.ae

Abstract

Neural Marked Temporal Point Processes (MTPP) are flexible models typically trained from large collections of sequences of irregularly spaced labeled events. These models inherently learn two predictive distributions: one for the arrival times of events and another for the types of events, also known as marks. In this study, we demonstrate that learning an MTPP model can be framed as a two-task learning problem, where both tasks share a common set of trainable parameters that are optimized jointly. We show that this practice can lead to conflicting gradients during training, resulting in overall degraded performance for both tasks. To overcome this issue, we introduce novel parametrizations for neural MTPP models that allow for separate modeling and training of each task, effectively avoiding the problem of conflicting gradients.

1 Introduction

Sequences of labeled events observed at irregular intervals in continuous time are ubiquitous across various fields such as healthcare [17], finance [30], social media [18], and seismology [52]. In numerous application domains, an important problem involves predicting the timing and types of future events—often called marks—based on historical data. Temporal Point Processes (TPPs) [13] provide a mathematical framework for modeling sequences of irregularly-spaced events, enabling subsequent inference on the system's evolution. Inspired by the success of foundation models (FMs) in natural language processing (NLP) [54, 14, 39, 8], FMs have been developed for time series forecasting, demonstrating strong generalization capabilities [3, 74, 56, 22]. Similarly, to address the limitations of classical temporal point process (TPP) models like the Hawkes process [29], recent efforts have introduced a more flexible class of neural marked TPP (MTPP) models, which leverage advances in deep learning [61], particularly through RNN [60, 78, 28] and Transformer [80, 77, 73] architectures.

This paper argues that learning a neural MTPP model can be interpreted as a two-task learning problem where both tasks share a common set of parameters and are optimized jointly. We identify these tasks as time prediction and mark prediction, respectively. While parameter sharing between tasks can sometimes enhance training efficiency [65], it may also result in performance degradation when compared to training each task separately. A major challenge in the simultaneous optimization of multi-task objectives is the issue of conflicting gradients [43]. When such conflicts arise, gradient updates tend to favor tasks with larger gradient magnitudes, thus hindering the learning process of other concurrent tasks and adversely affecting their performance. Although the phenomenon of conflicting gradients has been studied in various fields [10, 11, 75, 63], its impact on the training of neural MTPP models for irregular time series data remains unexplored. In this study, we demonstrate that conflicting gradients frequently occur during the training of neural MTPP models. Furthermore, we show that such conflicts can significantly degrade a model's predictive performance. To prevent

NeurIPS 2024 Workshop on Time Series in the Age of Large Models.

this issue, we introduce novel parametrizations for existing neural MTPP models, allowing for separate modeling and training of time and mark prediction tasks.

2 A Framework to Prevent Conflicting Gradients in Neural MTPP Models

A **Marked Temporal Point Process** (MTPP) is a random process whose realization is a sequence of events $\mathcal{S} = \{e_i = (t_i, k_i)\}_{i=1}^n$. Each event $e_i \in \mathcal{S}$ is an ordered pair with an *arrival time* $t_i \in [0, T]$ (with $t_0 = 0$) and a categorical label $k_i \in \mathbb{K} = \{1, ..., K\}$ called *mark*. The arrival times form a sequence of strictly increasing random values observed within a specified time interval $[0, T]$, i.e. $0 \leq t_1 < t_2 < \ldots < t_n \leq T$. If e_{i-1} is the last observed event, the occurrence of the next event e_i in $(t_{i-1}, \infty[$ can be fully characterized by its K conditional *marked intensity functions* $\lambda_k(t|\mathcal{H}_t)$ [55], where $\mathcal{H}_t = \{(t_i, k_i) \in \mathcal{S} \mid t_i < t\}$ is the observed process history. For clarity of notations, we will use the notation '$*$' of [13] to indicate dependence on $\mathcal{H}_t$, i.e. $\lambda_k^*(t) = \lambda^*(t|\mathcal{H}_t)$. The marked intensities can be further factorized as $\lambda_k^*(t) = \lambda^*(t)p^*(k|t)$ where $\lambda^*(t)$ is the *ground intensity* of the process, and $p^*(k|t)$ is a conditional PMF of marks. Finally, we can also define the marked *compensators* $\Lambda_k^*(t) = \int_{t_{i-1}}^t \lambda_k^*(s)ds$. Provided that certain modeling constraints are satisfied, each of $\lambda_k^*(t)$ and $\Lambda_k^*(t)$ fully characterizes a MTPP.

Let $\lambda_k^*(t; \boldsymbol{\theta}) = \lambda^*(t; \boldsymbol{\theta})p^*(k|t; \boldsymbol{\theta})$ be a valid model of $\lambda_k^*(t)$ (i.e. it defines a valid probability distribution of arrival times and marks), where $\boldsymbol{\theta}$ denotes the set of trainable parameters. To train this model, we use a dataset $\mathcal{S}_{\text{train}} = \{\mathcal{S}_1, ..., \mathcal{S}_L\}$, where each sequence $\mathcal{S}_l$ comprises n_l events with arrival times observed within the interval $[0, T]$ and $l = 1, ..., L$. The training objective is the average sequence negative log-likelihood (NLL) given by

$$\mathcal{L}(\boldsymbol{\theta}; \mathcal{S}_{\text{train}}) = \underbrace{-\frac{1}{L}\sum_{l=1}^{L}\sum_{i=1}^{n_l} \log \lambda^*(t_{l,i}; \boldsymbol{\theta}) + \int_0^T \lambda^*(s; \boldsymbol{\theta})ds}_{\mathcal{L}_T(\boldsymbol{\theta}; \mathcal{S}_{\text{train}})} \underbrace{-\frac{1}{L}\sum_{l=1}^{L}\sum_{i=1}^{n_l} \log p^*(k_{l,i}|t_{l,i}; \boldsymbol{\theta})}_{\mathcal{L}_M(\boldsymbol{\theta}; \mathcal{S}_{\text{train}})}, \quad (1)$$

This reveals that learning a MTPP model effectively involves a two-task learning problem with shared parameters $\boldsymbol{\theta}$. The time prediction task $\mathcal{T}_T$ focuses on learning the predictive ground intensity $\lambda^*(t; \boldsymbol{\theta})$ by minimizing $\mathcal{L}_T(\boldsymbol{\theta}, \mathcal{S}_{\text{train}})$. Concurrently, the mark prediction task $\mathcal{T}_M$ aims to learn the predictive distribution of marks $p^*(k|t; \boldsymbol{\theta})$ by minimizing $\mathcal{L}_M(\boldsymbol{\theta}, \mathcal{S}_{\text{train}})$.

Conflicting gradients. Let $\boldsymbol{g}_T = \nabla_{\boldsymbol{\theta}}\mathcal{L}_T(\boldsymbol{\theta})$ and $\boldsymbol{g}_M = \nabla_{\boldsymbol{\theta}}\mathcal{L}_M(\boldsymbol{\theta})$ denote the gradients of tasks $\mathcal{T}_T$ and $\mathcal{T}_M$, respectively, with respect to the shared parameters $\boldsymbol{\theta}$[1]. As discussed in [63], when $\boldsymbol{g}_T$ and $\boldsymbol{g}_M$ are pointing in opposite directions, i.e. $\boldsymbol{g}_T \cdot \boldsymbol{g}_M < 0$, an update step in the direction of negative $\boldsymbol{g}_T$ for $\boldsymbol{\theta}$ will increase the loss for task $\mathcal{T}_M$, and inversely for task $\mathcal{T}_T$ if an update step is taken in the direction of negative $\boldsymbol{g}_M$. Such *conflicting gradients* can be formally defined as follows.

Definition 1 (Conflicting gradients [63]) *Let ϕ_{TM} be the angle between the gradients $\boldsymbol{g}_T$ and $\boldsymbol{g}_M$. They are said to be conflicting with each other if* $\cos \phi_{TM} < 0$.

The smaller the value of $\cos \phi_{TM} \in [-1, 1]$, the more severe the conflict between the gradients. Conflicting gradients, especially those with significant differences in magnitude, pose substantial challenges during the optimization of multi-task learning objectives [75]. Specifically, if $\boldsymbol{g}_T$ and $\boldsymbol{g}_M$ conflict, the update step for $\boldsymbol{\theta}$ will likely be dominated by the gradient of whichever task—$\boldsymbol{g}_T$ or $\boldsymbol{g}_M$—has the greater magnitude, thereby disadvantaging the other task.

2.1 Disjoint Parametrizations of Neural MTPP models

Our goal is to prevent the occurrence of conflicting gradients during the training of neural MTPP models with the NLL objective given in (1). To accomplish this, we introduce novel parametrizations for neural MTPP models that enable disjoint modeling and training of time and mark prediction tasks. To accomplish this, we need to specify $p^*(k|t; \boldsymbol{\theta}_p)$ with parameters $\boldsymbol{\theta}_p$, and either $\lambda^*(t; \boldsymbol{\theta}_\lambda)$, or $\Lambda^*(t; \boldsymbol{\theta}_\lambda)$ with parameters $\boldsymbol{\theta}_\lambda$. For a query time $t \geq t_{i-1}$, we also define a history representation $\boldsymbol{h} = \text{ENC}(\{\boldsymbol{e}_1, ..., \boldsymbol{e}_{i-1}\}; \boldsymbol{\theta}_h) \in \mathbb{R}^{d_h}$, where $\text{ENC}(\cdot\,; \boldsymbol{\theta}_h)$ denotes the history encoder with parameters $\boldsymbol{\theta}_h$, and $\boldsymbol{e}_i \in \mathbb{R}^{d_e}$ is an encoding of event $e_i = (t_i, k_i)$. More training details are given in Appendix A.2, and the original model formulations are provided in Appendix A.5.

[1]We explicitly omit the dependency of $\mathcal{L}_T$ and $\mathcal{L}_M$ on $\mathcal{S}_{\text{train}}$ to simplify notations.

A general approach to model the distribution of marks. Given a query time $t \geq t_{i-1}$ and its corresponding history representation $\boldsymbol{h}$, we propose defining the mark conditional PMF $p^*(k|t;\boldsymbol{\theta}_p)$ using the following simple model:

$$p^*(k|t;\boldsymbol{\theta}_p) = \sigma_{So}\left(\mathbf{W}_2\sigma_R\left(\mathbf{W}_1\left[\boldsymbol{h}||\log(\tau)\right]+\boldsymbol{b}_1\right)\right)+\boldsymbol{b}_2\right), \tag{2}$$

where $\tau = t - t_{i-1}$, σ_{So} is the Softmax activation function, $\mathbf{W}_1 \in \mathbb{R}^{d_1\times(d_h+1)}$, $\boldsymbol{b}_1 \in \mathbb{R}^{d_1}$, $\mathbf{W}_2 \in \mathbb{R}^{K\times d_1}$ and $\boldsymbol{b}_2 \in \mathbb{R}^K$, and $||$ means concatenation. Here, $\boldsymbol{\theta}_p = \{\mathbf{W}_1, \mathbf{W}_2, \boldsymbol{b}_1, \boldsymbol{b}_2, \boldsymbol{\theta}_h\}$. Despite its simplicity, this model is flexible and capable of capturing the evolving dynamics of the mark distribution between two events.

Intensity-based parametrizations. We first consider MTPP models specified by their marked intensities, namely THP [80] and SAHP [77]. We propose revising the original model formulations to directly parametrize $\lambda^*(t)$, while $p^*(k|t)$ is systematically derived from expression (2).

SAHP++. While the original model formulation parametrizes $\lambda_k^*(t)$, we adapt its formulation to define:

$$\lambda^*(t;\boldsymbol{\theta}_\lambda) = \mathbf{1}^\mathsf{T}\left[\sigma_S\left(\boldsymbol{\mu}-(\boldsymbol{\eta}-\boldsymbol{\mu})\exp(-\boldsymbol{\gamma}(t-t_{i-1}))\right)\right], \tag{3}$$

where $\mathbf{1} \in \mathbb{R}^C$ is a vector of 1's allowing to define $\lambda^*(t;\boldsymbol{\theta}_\lambda)$ as a sum over C different representations. In (3), $\boldsymbol{\mu} = \sigma_G(\mathbf{W}_\mu\boldsymbol{h})$, $\boldsymbol{\eta} = \sigma_S(\mathbf{W}_\eta\boldsymbol{h})$ and $\boldsymbol{\gamma} = \sigma_G(\mathbf{W}_\gamma\boldsymbol{h})$ where σ_S and σ_G are respectively the softplus and GeLU activation functions [31]. $\mathbf{W}_\mu, \mathbf{W}_\eta, \mathbf{W}_\gamma \in \mathbb{R}^{C\times d_h}$ are learnable parameters and $\boldsymbol{\theta}_\lambda = \{\mathbf{W}_\mu, \mathbf{W}_\eta, \mathbf{W}_\gamma, \boldsymbol{\theta}_h\}$.

THP++. Following a similar reasoning, the original formulation of THP is adapted to model $\lambda^*(t)$ instead of $\lambda_k^*(t)$:

$$\lambda^*(t;\boldsymbol{\theta}_\lambda) = \mathbf{1}^\mathsf{T}\left[\sigma_S\left(\boldsymbol{w}_t\frac{t-t_{i-1}}{t_{i-1}}+\mathbf{W}\boldsymbol{h}+\boldsymbol{b}\right)\right], \tag{4}$$

where $\boldsymbol{w}_t \in \mathbb{R}_+^C$, $\mathbf{W} \in \mathbb{R}^{C\times d_h}$, $\boldsymbol{b} \in \mathbb{R}^{d_1}$, and $\boldsymbol{\theta}_\lambda = \{\mathbf{W}, \boldsymbol{w}_t, \boldsymbol{b}, \boldsymbol{\theta}_h\}$.

Cumulative intensity-based parametrizations. Integrating cumulative neural MTPP models into our framework requires to define $\Lambda^*(t;\boldsymbol{\theta}_\Lambda)$. Specifically, we extend the improved marked FullyNN model (FNN) [53, 17] into *FNN*$^+$, that models $\Lambda^*(t)$ and $p^*(k|t)$, instead of $\Lambda_k^*(t)$:

$$\Lambda^*(t;\boldsymbol{\theta}_\lambda) = G^*(t;\boldsymbol{\theta}_\lambda) - G^*(t_{i-1};\boldsymbol{\theta}_\lambda), \tag{5}$$

$$G^*(t;\boldsymbol{\theta}_\lambda) = \mathbf{1}^\mathsf{T}\left[\sigma_S\big(\mathbf{W}(\sigma_{GS}\big(\boldsymbol{w}_t(t-t_{i-1})+\mathbf{W}_h\boldsymbol{h}+\boldsymbol{b}_1\big)+\boldsymbol{b}_2\big)\right], \tag{6}$$

where $\mathbf{W} \in \mathbb{R}^{C\times d_1}$, $\boldsymbol{w}_t, \boldsymbol{b}_1 \in \mathbb{R}^{d_1}$, $\mathbf{W}_h \in \mathbb{R}^{d_1\times d_h}$, $\boldsymbol{b}_2 \in \mathbb{R}^C$, and σ_{GS} is the Gumbel-Softplus activation function. Here, $\boldsymbol{\theta}_\lambda = \{\mathbf{W}, \mathbf{W}_h, \boldsymbol{w}_t, \boldsymbol{b}_1, \boldsymbol{b}_2, \boldsymbol{\theta}_h\}$. Similarly to the previous models, we use (2) to define $p^*(k|t;\boldsymbol{\theta}_p)$.

Training different history encoders. The different functions defined in this section, $p^*(k|\tau;\boldsymbol{\theta}_p)$, $\lambda^*(t;\boldsymbol{\theta}_\lambda)$, and $\Lambda^*(t;\boldsymbol{\theta}_\lambda)$, share a common set of parameters $\boldsymbol{\theta}_h$ through a common history representation $\boldsymbol{h}$. To enable fully disjoint modeling and training of time and mark functions, we define two distinct history representations:

$$\boldsymbol{h}^t = \text{ENC}_t\left[\{\boldsymbol{e}_1,...,\boldsymbol{e}_{i-1}\};\boldsymbol{\theta}_h^t\right] \in \mathbb{R}^{d_h^t} \quad \text{and} \quad \boldsymbol{h}^m = \text{ENC}_m\left[\{\boldsymbol{e}_1,...,\boldsymbol{e}_{i-1}\};\boldsymbol{\theta}_h^m\right] \in \mathbb{R}^{d_h^m}, \tag{7}$$

where $\text{ENC}_t(\cdot\,;\boldsymbol{\theta}_h^t)$ and $\text{ENC}_m(\cdot\,;\boldsymbol{\theta}_h^m)$ are the *time* and *mark* history encoders, respectively, while $\boldsymbol{\theta}_h^t$ and $\boldsymbol{\theta}_h^m$ represent the sets of *disjoint* learnable parameters of the two encoders. Using separate history encoders further enables the model to capture information from past event occurrences that are relevant to the time and mark prediction tasks separately. In this paper, without loss of generality, we compute $\boldsymbol{h}^t$ and $\boldsymbol{h}^m$ by training two GRU encoders that sequentially process the set of event representations in $\{\boldsymbol{e}_1,...,\boldsymbol{e}_{i-1}\}$.

Disjoint training of the time and mark tasks. Using $\boldsymbol{h}^t$ for $\lambda^*(t;\boldsymbol{\theta}_\lambda)$ and $\Lambda^*(t;\boldsymbol{\theta}_\lambda)$, and $\boldsymbol{h}^m$ for $p^*(k|\tau;\boldsymbol{\theta}_p)$[2] implies that $\boldsymbol{\theta}_\lambda$ and $\boldsymbol{\theta}_p$ are now completely disjoint set of trainable parameters. Consequently, by injecting $\lambda^*(t;\boldsymbol{\theta}_\lambda)$ or $d\Lambda^*(t;\boldsymbol{\theta}_\lambda)/dt$, and $p^*(k|t;\boldsymbol{\theta}_p)$ in (1), we find that the NLL objective now writes as sum over two entirely disjoint objectives $\mathcal{L}_T(\boldsymbol{\theta}_\lambda, \mathcal{S}_{\text{train}})$ and $\mathcal{L}_M(\boldsymbol{\theta}_p, \mathcal{S}_{\text{train}})$, meaning that the associated tasks $\mathcal{T}_T$ and $\mathcal{T}_M$ can be learned independently. This contrasts with the original formulations of THP, SAHP, and FNN [53, 17, 80, 77], in which shared parameters between $\mathcal{L}_T(\boldsymbol{\theta}_\lambda, \mathcal{S}_{\text{train}})$ and $\mathcal{L}_M(\boldsymbol{\theta}_p, \mathcal{S}_{\text{train}})$ does not allow for disjoint training of (1).

[2] $\boldsymbol{\theta}_h$ is now replaced by $\boldsymbol{\theta}_h^t$ in $\boldsymbol{\theta}_\lambda$, and $\boldsymbol{\theta}_\Lambda$, and by $\boldsymbol{\theta}_h^m$ in $\boldsymbol{\theta}_p$.

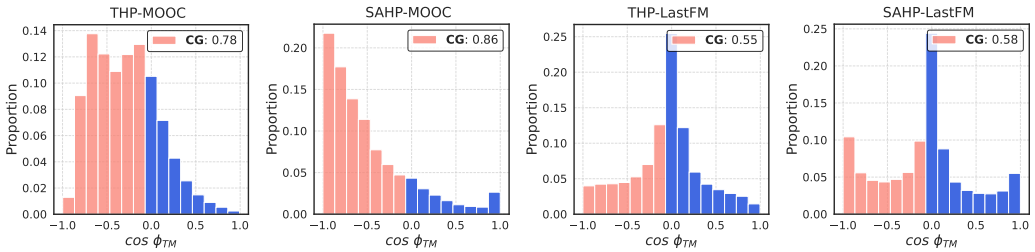


Figure 1: Distribution of cos ϕ_{TM} during training of THP and SAHP on MOOC and LastFM. CG refers to the proportion of cos $\phi_{TM} < 0$ (red bars) observed during training. The distribution is obtained by pooling the values of ϕ_{TM} over 3 training runs.

3 Experiments

Experimental setup. We conduct an experimental study to assess the performance of our framework as detailed in training the time and mark prediction tasks. Specifically, we compare the disjoint parametrizations of **THP++**, **SAHP++**, and **FNN++** detailed in Section 2 compared to their original formulations[3]. To this end, we use five real-world marked event sequence datasets frequently referenced in the neural TPP literature: **LastFM**, **MOOC**, **Reddit** [38], **Github** [68], and **Stack Overflow** [16]. To ensure that performance gains cannot be attributed to increased model capacity, we ensure that the number of parameters between the base and base**++** parametrizations remains equivalent. See Appendix A.2 for more details on the training setup, and Appendix A.3 for further dataset descriptions. To evaluate the performance of the different baselines on the time and mark prediction tasks, we report the $\mathcal{L}_T$ and $\mathcal{L}_M$ terms in (1), respectively, computed over all test sequences.

Preventing conflicts lead to improved performance. We present the $\mathcal{L}_T$ and $\mathcal{L}_M$ metrics for the base and base**++** models across all datasets in Table 1. We observe a consistent improvement for THP**++**, SAHP**++** and FNN**++** compared to their original parametrizations. Figure 1 shows the distribution of cos ϕ_{TM} during training for THP and SAHP on LastFM and MOOC. As observed, a significant proportion of severe conflicts (as indicated by cos ϕ_{TM} in [-1, -0.5]) emerge for the shared parameters of the base models during training. Combined with the $\mathcal{L}_T$ and $\mathcal{L}_M$ metrics from Table 1, this observation highlights the advantages of our disjoint parametrization frameworks in preventing conflicting gradients between the time and mark prediction tasks during training. We report results with respect to other evaluation metrics in Appendix A.7.

Table 1: $\mathcal{L}_T$ and $\mathcal{L}_M$ results of the different setups across all datasets. The values are computed over 3 splits, and the standard error is reported in parenthesis. Best results are highlighted in bold.

	$\mathcal{L}_M$				
	LastFM	MOOC	Github	Reddit	Stack O.
THP	714.3 (18.1)	93.5 (1.8)	133.2 (28.5)	**40.7 (1.4)**	104.9 (1.1)
THP++	**647.0 (20.2)**	**71.1 (1.4)**	**117.6 (25.0)**	41.2 (1.6)	**102.7 (1.1)**
SAHP	829.4 (28.3)	162.8 (3.6)	144.3 (31.2)	73.2 (5.7)	106.9 (0.6)
SAHP++	**651.3 (18.7)**	**71.0 (1.5)**	**118.5 (25.0)**	**41.2 (1.0)**	**102.9 (1.1)**
FNN	739.3 (30.1)	79.3 (1.8)	119.1 (25.1)	45.7 (1.2)	107.0 (1.0)
FNN++	**647.9 (17.6)**	**71.8 (1.5)**	**114.9 (24.4)**	**40.6 (1.0)**	**102.8 (1.1)**

	$\mathcal{L}_T$				
	LastFM	MOOC	Github	Reddit	Stack O.
THP	-961.3 (44.0)	**-136.7 (1.8)**	-258.8 (62.4)	-71.7 (3.4)	-82.9 (2.1)
THP++	**-1052.8 (50.1)**	-135.6 (2.8)	**-297.3 (80.9)**	**-87.5 (2.9)**	**-83.1 (2.1)**
SAHP	-1266.9 (64.7)	-266.5 (4.3)	-380.2 (89.8)	-74.0 (2.6)	**-88.7 (2.2)**
SAHP++	**-1322.6 (67.9)**	**-295.3 (4.8)**	**-400.8 (92.6)**	**-94.8 (3.2)**	-76.6 (1.7)
FNN	-1271.9 (65.6)	-281.9 (4.1)	-395.4 (89.9)	-76.8 (2.8)	-80.2 (2.0)
FNN++	**-1321.1 (62.8)**	**-301.9 (4.7)**	**-395.5 (89.6)**	**-95.6 (3.3)**	**-87.7 (2.1)**

4 Conclusion, Limitations, and Future Work

In this paper, we demonstrate that shared parameters between the time and mark prediction tasks in neural MTPP models lead to the emergence of conflicting gradients during training, often resulting in degraded performance on each individual task. To tackle this issue, we introduce novel parametrizations of neural MTPP models that enable separate training and modeling of each task, effectively preventing the occurrence of conflicting gradients. Through extensive experiments on real-world event sequence datasets, we validate the benefits of our disjoint training framework over original model parametrizations. Nonetheless, we acknowledge that our study has limitations: it focuses only on categorical marks and models trained with the NLL objective. Future research could pertain to exploring conflicting gradients using alternative scoring rules and in more more complex scenarios, such as temporal graphs or spatio-temporal processes.

[3]For the remainder of this paper, THP, SAHP and FNN will be referred to as 'base models', while THP**++**, SAHP**++** and FNN**++** will be referred to as 'base**++**' models.

References

[1] Souhaib Ben Taieb. Learning quantile functions for temporal point processes with recurrent neural splines, 2022. AISTATS.

[2] Marin Biloš, Bertrand Charpentier, and Stephan Günnemann. Uncertainty on asynchronous time event prediction, 2020. Neurips.

[3] Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, Erik Brynjolfsson, Shyamal Buch, Dallas Card, Rodrigo Castellon, Niladri Chatterji, Annie Chen, Kathleen Creel, Jared Quincy Davis, Dora Demszky, Chris Donahue, Moussa Doumbouya, Esin Durmus, Stefano Ermon, John Etchemendy, Kawin Ethayarajh, Li Fei-Fei, Chelsea Finn, Trevor Gale, Lauren Gillespie, Karan Goel, Noah Goodman, Shelby Grossman, Neel Guha, Tatsunori Hashimoto, Peter Henderson, John Hewitt, Daniel E. Ho, Jenny Hong, Kyle Hsu, Jing Huang, Thomas Icard, Saahil Jain, Dan Jurafsky, Pratyusha Kalluri, Siddharth Karamcheti, Geoff Keeling, Fereshte Khani, Omar Khattab, Pang Wei Koh, Mark Krass, Ranjay Krishna, Rohith Kuditipudi, Ananya Kumar, Faisal Ladhak, Mina Lee, Tony Lee, Jure Leskovec, Isabelle Levent, Xiang Lisa Li, Xuechen Li, Tengyu Ma, Ali Malik, Christopher D. Manning, Suvir Mirchandani, Eric Mitchell, Zanele Munyikwa, Suraj Nair, Avanika Narayan, Deepak Narayanan, Ben Newman, Allen Nie, Juan Carlos Niebles, Hamed Nilforoshan, Julian Nyarko, Giray Ogut, Laurel Orr, Isabel Papadimitriou, Joon Sung Park, Chris Piech, Eva Portelance, Christopher Potts, Aditi Raghunathan, Rob Reich, Hongyu Ren, Frieda Rong, Yusuf Roohani, Camilo Ruiz, Jack Ryan, Christopher Ré, Dorsa Sadigh, Shiori Sagawa, Keshav Santhanam, Andy Shih, Krishnan Srinivasan, Alex Tamkin, Rohan Taori, Armin W. Thomas, Florian Tramèr, Rose E. Wang, William Wang, Bohan Wu, Jiajun Wu, Yuhuai Wu, Sang Michael Xie, Michihiro Yasunaga, Jiaxuan You, Matei Zaharia, Michael Zhang, Tianyi Zhang, Xikun Zhang, Yuhui Zhang, Lucia Zheng, Kaitlyn Zhou, and Percy Liang. On the opportunities and risks of foundation models. *ArXiv preprint*, 2022.

[4] Tanguy Bosser and Souhaib Ben Taieb. On the predictive accuracy of neural temporal point process models for continuous-time event data, 2023. TMLR.

[5] Felix J. S. Bragman, Ryutaro Tanno, Sebastien Ourselin, Daniel C. Alexander, and M. Jorge Cardoso. Stochastic filter groups for multi-task cnns: Learning specialist and generalist convolution kernels, 2019. Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision (ICCV).

[6] Jonas Brehmer, Tilmann Gneiting, Marcus Herrmann, Warner Marzocchi, Martin Schlather, and Kirstin Strokorb. Comparative evaluation of point process forecasts, 2021.

[7] Glenn W. Brier. Verification of forecasts expressed in terms of probability, 1950. Monthly Weather Review.

[8] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *Proceedings of the 34th Conference on Neural Information Processing Systems*, 2020.

[9] David Bruggemann, Menelaos Kanakis, Stamatios Georgoulis, and Luc Van Gool. Automated search for resource-efficient branched multi-task networks, 2020. Proceedings of the 31st British Machine Vision Conference (BMVC).

[10] Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks, 2018. ICML.

[11] Zhao Chen, Jiquan Ngiam, Yanping Huang, Thang Luong, Henrik Kretzschmar, Yuning Chai, and Dragomir Anguelov. Just pick a sign: Optimizing deep multitask models with gradient sign dropout, 2020. Proceedings of the 34th International Conference on Neural Information Processing Systems (NeurIPS),.

[12] Nick Craswell. Mean reciprocal rank, 2009. Encyclopedia of Database Systems, Springer.

[13] Daryl J. Daley and David Vere-Jones. An introduction to the theory of point processes. volume ii: General theory and structure. *Springer*, 2008.

[14] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, 2019.

[15] Victor Dheur and Souhaib Ben Taieb. A large-scale study of probabilistic calibration in neural network regression, 2023. ICML.

[16] Nan Du, Hanjun Dai, Rakshit Trivedi, Utkarsh Upadhyay, Manuel Gomez-Rodriguez, and Le Song. Recurrent marked temporal point processes: Embedding event history to vector, 2016. SIGKDD.

[17] Joseph Enguehard, Dan Busbridge, Adam Bozson, Claire Woodcock, and Nils Y. Hammerla. Neural temporal point processes for modelling electronic health records, 2020. ML4H.

[18] Mehrdad Farajtabar, Yichen Wang, Manuel Gomez Rodriguez, Shuang Li, Hongyuan Zha, and Le Song. Coevolve: A joint point process model for information diffusion and network co-evolution, 2017. Journal of Machine Learning Research, 18, 1-49.

[19] Christopher Fifty, Ehsan Amid, Zhe Zhao, Tianhe Yu, Rohan Anil, and Chelsea Finn. Efficiently identifying task groupings for multi-task learning, 2021. Proceedings of the 35th International Conference on Neural Information Processing Systems (NeurIPS).

[20] Yuan Gao, Haoping Bai, Zequn Jie, Jiayi Ma, Kui Jia, and Wei Liu. Mtl-nas: Task-agnostic neural architecture search towards general-purpose multi-task learning, 2020. Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition.

[21] Yuan Gao, Jiayi Ma, Mingbo Zhao, Wei Liu, and Alan L. Yuille. Nddr-cnn: Layerwise feature fusing in multi-task cnns by neural discriminative dimensionality reduction, 2019. Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition.

[22] Azul Garza, Cristian Challu, and Max Mergenthaler-Canseco. Timegpt-1. *ArXiv preprint*, 2024.

[23] Tilmann Gneiting, Fadoua Balabdaoui, and Adrian E. Raftery. Probabilistic forecasts, calibration and sharpness, 2007. Journal of the Royal Statistical Society: Series B (Statistical Methodology).

[24] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks, 2017. ICML.

[25] Michelle Guo, Albert Haque, De-An Huang, Serena Yeung, and Li Fei-Fei. Dynamic task prioritization for multitask learning, 2018. Proceedings of the European Conference on Computer Vision (ECCV).

[26] Pengsheng Guo, Chen-Yu Lee, and Daniel Ulbricht. Learning to branch for multi-task learning, 2020. Proceedings of the 37th International Conference on Machine Learning (ICML).

[27] Ruocheng Guo, Jundong Li, and Huan Liu. Initiator: Noise-contrastive estimation for marked temporal point process, 2018. IJCAI.

[28] Vinayak Gupta, Srikanta J. Bedathur, Sourangshu Bhattacharya, and A. De. Learning temporal point processes with intermittent observations, 2021. Proceedings of The 24th International Conference on Artificial Intelligence and Statistics (AISTATS).

[29] Alan G Hawkes. Point spectra of some mutually exciting point processes, 1971. Journal of the Royal Statistical Society: Series B, 33(3).

[30] Alan G. Hawkes. Hawkes processes and their applications to finance: a review, 2018. Quantitative Finance, 18(2), 193-198.

[31] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus), 2023.

[32] Valerie Isham and Mark Westcott. A self-correcting point process., 1979. Stochastic Processes and Their Applications, 8(3):335–347.

[33] Adrián Javaloy and Isabel Valera. Rotograd: Gradient homogenization in multitask learning, 2022. International Conference on Learning Representations (ICLR).

[34] Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics, 2018. Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition.

[35] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014. ICLR.

[36] Iasonas Kokkinos. Ubernet: Training a 'universal' convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory, 2017. Proceedings of the 2017 IEEE International Conference on Computer Vision and Pattern Recognition (CVPR).

[37] Volodymyr Kuleshov, Nathan Fenner, and Stefano Ermon. Accurate uncertainties for deep learning using calibrated regression, 2018. Proceedings of the 35 th International Conference on Machine Learning (ICML).

[38] Srijan Kumar, Xikun Zhang, and Jure Leskovec. Predicting dynamic embedding trajectory in temporal interaction networks. In *Proceedings of the 25th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2019. KDD.

[39] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020.

[40] Zichong Li, Yanbo Xu, Simiao Zuo, Haoming Jiang, Chao Zhang, Tuo Zhao, and Hongyuan Zha. Smurf-thp: Score matching-based uncertainty quantification for transformer hawkes process, 2023. ICML.

[41] Haitao Lin, Cheng Tan, Lirong Wu, Zhangyang Gao, and Stan. Z. Li. An empirical study: Extensive deep temporal point process, 2021.

[42] Haitao Lin, Lirong Wu, Guojiang Zhao, Pai Liu, and Stan Z. Li. Exploring generative neural temporal point process, 2022. TMLR.

[43] Bo Liu, Xingchao Liu, Xiaojie Jin, Peter Stone, and Qiang Liu. Conflict-averse gradient descent for multi-task learning, 2021. Neurips.

[44] Bo Liu, Xingchao Liu, Xiaojie Jin, Peter Stone, and Qiang Liu. Conflict-averse gradient descent for multi-task learning, 2021. Proceedings of the 35th International Conference on Neural Information Processing Systems (NeurIPS).

[45] Liyang Liu, Yi Li, Zhanghui Kuang, Jing-Hao Xue, Yimin Chen, Wenming Yang, Q. Liao, and Wayne Zhang. Towards impartial multi-task learning, 2021. International Conference on Learning Representations (ICLR).

[46] Shikun Liu, Edward Johns, and Andrew J. Davison. End-to-end multi-task learning with attention, 2019. Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).

[47] Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Philip S. Yu. Learning multiple tasks with multilinear relationship networks, 2017. Proceedings of the 31st International Conference on Neural Information Processing Systems (NeurIPS).

[48] Kevis-Kokitsi Maninis, Ilija Radosavovic, and Iasonas Kokkinos. Attentive single-tasking of multiple tasks, 2019. Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).

[49] Hongyuan Mei and Jason Eisner. The neural hawkes process: A neurally self-modulating multivariate point process, 2016. Neurips.

[50] Ishan Misra, Abhinav Shrivastava, Abhinav Gupta, and Martial Hebert. Cross-stitch networks for multi-task learning, 2016. Proceedings of the 2016 IEEE/CVF International Conference on Computer Vision and Pattern Recognition (CVPR).

[51] Mahdi Pakdaman Naeini, Gregory F. Cooper, and Milos Hauskrecht. Obtaining well calibrated probabilities using bayesian binning, 2015. AAAI.

[52] Yosihiko Ogata. Space-time point-process models for earthquake occurrences, 1998. Annals of the Institute of Statistical Mathematics, 50(2):379–402.

[53] Takahiro Omi, Naonori Ueda, and Kazuyuki Aihara. Fully neural network based model for general temporal point processes. 2019. Neurips.

[54] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. *Preprint*, 2018.

[55] Jakob Gulddahl Rasmussen. Lecture notes: Temporal point processes and the conditional intensity function, 2018.

[56] Kashif Rasul, Arjun Ashok, Andrew Robert Williams, Hena Ghonia, Rishika Bhagwatkar, Arian Khorasani, Mohammad Javad Darvishi Bayazi, George Adamopoulos, Roland Riachi, Nadhir Hassen, Marin Biloš, Sahil Garg, Anderson Schneider, Nicolas Chapados, Alexandre Drouin, Valentina Zantedeschi, Yuriy Nevmyvaka, and Irina Rish. Lag-llama: Towards foundation models for probabilistic time series forecasting. *R0-FoMo: Workshop on Robustness of Few-shot and Zero-shot Learning in Foundation Models at NeurIPS 2023.*, 2024.

[57] Sebastian Ruder, Joachim Bingel, Isabelle Augenstein, and Anders Sogaard. Latent multi-task architecture learning, 2019. Proceedings of the AAAI Conference on Artificial Intelligence.

[58] Ozan Sener and Vladlen Koltun. Multi-task learning as multi-objective optimization, 2018. Proceedings of the 32nd International Conference on Neural Information Processing Systems (NeurIPS).

[59] Karishma Sharma, Yizhou Zhang, Emilio Ferrara, and Yan Liu. Identifying coordinated accounts on social media through hidden influence and group behaviours, 2021. Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining.

[60] Oleksandr Shchur, Marin Biloš, and Stephan Günnemann. Intensity-free learning of temporal point processes, 2020. ICLR.

[61] Oleksandr Shchur, Ali Caner Türkmen, Tim Januschowski, and Stephan Günnemann. Neural temporal point processes: A review, 2021. Proceedings of 13th Joint Conference on Artificial Intelligence.

[62] Jiayi Shen, Xiantong Zhen, Marcel Worring, and Ling Shao. Variational multi-task learning with gumbel-softmax priors, 2021. Proceedings of the 35th International Conference on Neural Information Processing Systems (NeurIPS).

[63] Guangyuan Shi, Qimai Li, Wenlong Zhang, Jiaxin Chen, and Xiao-Ming Wu. Recon: Reducing conflicting gradients from the root for multi-task learning, 2023. ICLR.

[64] Ayan Sinha, Zhao Chen, Vijay Badrinarayanan, and Andrew Rabinovich. Gradient adversarial training of neural networks, 2018.

[65] Trevor Standley, Amir R. Zamir, Dawn Chen, Leonidas Guibas, Jitendra Malik, and Silvio Savarese. Which tasks should be learned together in multi-task learning?, 2020. ICML.

[66] Trevor Standley, Amir R. Zamir, Dawn Chen, Leonidas Guibas, Jitendra Malik, and Silvio Savarese. Which tasks should be learned together in multi-task learning?, 2020. Proceedings of the 37th International Conference on Machine Learning (ICML).

[67] Sebastian Thrun and Joseph O'Sullivan. Discovering structure in multiple learning tasks: The tc algorithm, 1996. Proceedings of the 13th International Conference on Machine Learning (ICML).

[68] Rakshit Trivedi, Mehrdad Farajtabar, Prasenjeet Biswal, and Hongyuan Zha. Representation learning over dynamic graphs, 2019. International Conference on Learning Representations (ICLR).

[69] Ali Caner Türkmen, Bernie Wang, and Alex Smola. Fastpoint: Scalable deep point processes, 2019. Proceedings of European Conference on Machine Learning and Knowledge Discovery in Databases (ECML PKDD).

[70] Govind Waghmare, Ankur Debnath, Siddhartha Asthana, and Aakarsh Malhotra. Modeling inter-dependence between time and mark in multivariate temporal point processes, 2022. CIKM.

[71] Zirui Wang, Yulia Tsvetkov, Orhan Firat, and Yuan Cao. Gradient vaccine: Investigating and improving multi-task optimization in massively multilingual models, 2020. International Conference on Learning Representations (ICLR).

[72] Shuai Xiao, Junchi Yan, Stephen M. Chu, Xiaokang Yang, and Hongyuan Zha. Modeling the intensity function of point process via recurrent neural networks, 2017. Proceedings of the 31st AAAI Conference on Artificial Intelligence.

[73] Chenghao Yang, Hongyuan Mei, and Jason Eisner. Transformer embeddings of irregularly spaced events and their participants, 2022.

[74] Chin-Chia Michael Yeh, Xin Dai, Huiyuan Chen, Yan Zheng, Yujie Fan, Audrey Der, Vivian Lai, Zhongfang Zhuang, Junpeng Wang, Liang Wang, and Wei Zhang. Toward a foundation model for time series data. *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management (CIKM)*, 2023.

[75] Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning, 2020. Proceedings of the 34th International Conference on Neural Information Processing Systems (NeurIPS).

[76] Amir Zamir, Alexander Sax, William Shen, Leonidas Guibas, Jitendra Malik, and Silvio Savarese. Taskonomy: Disentangling task transfer learning, 2018. Proceedings of the 2018 IEEE/CVF International Conference on Computer Vision and Pattern Recognition (CVPR).

[77] Qiang Zhang, Aldo Lipani, Omer Kirnap, and Emine Yilmaz. Self-attentive hawkes processes, 2020. ICML.

[78] Shixiang Zhu, Henry Shaowu Yuchi, and Yao Xie. Adversarial anomaly detection for marked spatio-temporal streaming data, 2020. Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).

[79] Shixiang Zhu, Minghe Zhang, Ruyi Ding, and Yao Xie. Deep fourier kernel for self-attentive point processes, 2021. Proceedings of The 24th International Conference on Artificial Intelligence and Statistics (AISTATS).

[80] Simiao Zuo, Haoming Jiang, Zichong Li, Tuo Zhao, and Hongyuan Zha. Transformer hawkes process, 2020. ICML.

A Appendix

A.1 Related Work

Neural MTPP models. To address the limitations of simple parametric MTPP models [29, 32], prior studies focused on designing more flexible approaches by leveraging recent advances in deep learning. Based on the parametrization chosen, these neural MTPP models can be generally classified along three main axis: density-based, intensity-based, and compensator-based approaches. Intensity-based approaches propose to model the trajectories of future arrival-times and marks by parametrizing the marked intensities $\lambda_k^*(t)$. In this line of work, past event occurrences are usually encoded into a history representation using RNNs [16, 49, 27, 69, 2, 78] or self-attention (SA) mechanisms [80, 77, 79, 73, 40]. However, parametrizations of the marked intensity functions often come at the cost of being unable to evaluate the log-likelihood in closed-form, requiring Monte Carlo integration. This consideration motivated the design of compensator-based approaches that parametrize $\Lambda_k^*(t)$ using fully-connected neural networks [53], or SA mechanisms [17], from which $\lambda_k^*(t)$ can be retrieved through differentiation. Finally, density-based approaches aim at directly modeling the joint density of (inter-)arrival times and marks $f^*(\tau, k)$. Among these, different family of distributions have been considered to model the distribution of inter-arrival times [72, 41]. Notably [60] relies on a mixture of log-normal distributions to estimate $f^*(\tau)$, a model that then appeared in subsequent works [59, 28]. However, the original work of [60] assumes conditional independence of inter-arrival times and marks given the history, which is alleviated in [70]. Nonetheless, a common thread of these parametrizations is that they explicitly enforce parameter sharing between the time and mark prediction tasks. As we demonstrate, this often leads to the emergence of conflicting gradient during training, hindering model's performance. For an overview of neural TPP models, we refer the reader to the works of [61], [42] and [4].

Conflicting gradients in multi-task learning. Diverse approaches have been investigated in the literature to improve interactions between concurrent tasks in multi-task learning problems, thereby boosting performance for each task individually. In this context, a prominent line of work focuses on balancing the different tasks at hand through direct manipulation of their gradients. These manipulations either aim at alleviating the differences in gradient magnitudes between tasks [10, 58, 45], or the emergence of conflicts [64, 48, 75, 11, 71, 44, 33]. Alternative approaches to task balancing have been explored based on different criteria, such as task prioritization [25], uncertainty [34], or learning pace [46]. Another line of work refers to task clustering methods, which aim at identifying the tasks that should be learned jointly [67, 76, 66, 62, 19]. At the network level, multi-task learning methods can be partitioned into two main groups. Hard parameter sharing methods denotes methods that share common parameters between multiple tasks [36, 47, 5]. Conversely, soft sharing methods regroups methods where each task possess its own set of parameters, although a sharing mechanism enables communication across tasks [50, 57, 21, 20]. Our methodology relates more to branched architecture search approaches [26, 9, 63], where the aim is set on dynamically identifying which layers should or should not be shared between tasks based on a chosen criterion, e.g. the proportion of conflicting gradients.

A.2 Training details

Training details. For all models, we minimize the average NLL in (1) on the training sequences using mini-batch gradient descent with the Adam optimizer [35] and a learning rate of 10^{-3}. For the base models, an early-stopping protocol interrupts training if the model fails to show improvement in the total validation loss (i.e., $\mathcal{L}_T$ + $\mathcal{L}_T$) for 50 consecutive epochs. Conversely, in the base**++** setting, two distinct early-stopping protocols are implemented for the $\mathcal{L}_T$ and $\mathcal{L}_M$ terms, respectively. If one of these terms does not show improvement for 50 consecutive epochs, we freeze the parameters of the associated functions (e.g. $\boldsymbol{\theta}_\lambda$ for $\lambda^*(t; \boldsymbol{\theta}_\lambda)$) and allow the remaining term to continue training. Training is ultimately interrupted if both early-stopping criteria are met. Figure 2 presents a graphical representation of the base and base**++** setups.

In all setups, the optimization process can last for a maximum of 500 epochs, and we revert the model parameters to their state with the lowest validation loss after training. Finally, we evaluate the model by computing test metrics on the test sequences of each split. Our framework is implemented in a

unified codebase using PyTorch[4]. All models were trained on a machine equipped with an AMD Ryzen Threadripper PRO 3975WX CPU running at 4.1 GHz and a Nvidia RTX A4000 GPU.

Encoding past events. To obtain the encoding $\boldsymbol{e}_i \in \mathbb{R}^{d_e}$ of an event $e_i = (t_i, k_i)$ in $\mathcal{H}_t$, we follow the work of [17] by first mapping t_i to a vector of sinusoidal functions:

$$\boldsymbol{e}_i^t = \bigoplus_{j=0}^{d_t/2-1} \sin\left(\alpha_j t_i\right) \oplus \cos\left(\alpha_j t_i\right) \in \mathbb{R}^{d_t}, \tag{8}$$

where $\alpha_j \propto 1000^{\frac{-2j}{d_t}}$ and $\oplus$ is the concatenation operator. Then, a mark embedding $\boldsymbol{e}_i^k \in \mathbb{R}^{d_k}$ for k_i is generated as $\boldsymbol{e}_i^k = \mathbf{E}^k \boldsymbol{k}_i$, where $\mathbf{E}^k \in \mathbb{R}^{d_k \times K}$ is a learnable embedding matrix, and $\boldsymbol{k}_i \in \{0,1\}^K$ is the one-hot encoding of k_i. Finally, we obtain $\boldsymbol{e}_i$ through concatenation, i.e. $\boldsymbol{e}_i = [\boldsymbol{e}_i^t || \boldsymbol{e}_i^k]$.

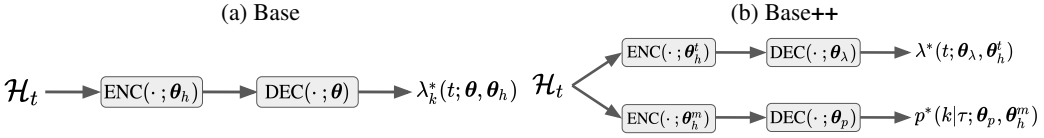


Figure 2: Graphical representation of the base and "**++**" setups.

Hyperparameters. To ensure that changes in performance are solely attributed to the features enabled by our framework, we control the number of parameters such the capacity of baselines remains equivalent between the base and base**++** settings. Table 2 provides the total number of trainable parameters for each setup when trained on the LastFM dataset, as well as their distribution across the encoder and decoder heads. For all baselines and setups, we use a single encoder layer, and the dimension d_e of the event encodings is set to 8. Additionally, we chose a value of $M = 32$ for the number of mixture components. It is worth noting that [60] found LogNormMix to be robust to the choice of M. Finally, we set the number of projections for $\lambda^*(t;\boldsymbol{\theta}_\lambda)$ to $C = 32$.

Table 2: Number of parameters for each baseline when trained on the LastFM dataset. The distribution of parameters between the encoder and decoder heads is reported in parenthesis.

	THP	SAHP	FNN
Base	14720 (0.66/0.34)	15588 (0.68/0.32)	15939 (0.65/0.35)
++	14602 (0.63/0.37)	15514 (0.67/0.33)	14961 (0.67/0.33)

A.3 Datasets

We use 5 real-world event sequence datasets for our experiments:

- **LastFM** [38]: Each sequence corresponds to a user listening to music records over time. The artist of the song is the mark.
- **MOOC** [38]: Records of students' activities on an online course system. Each sequence corresponds to a student, and the mark is the type of activity performed.
- **Github** [68]: Actions of software developers on the open-source platform Github. Each sequence corresponds a developer, and the mark is the action performed (e.g. fork, pull request,...).
- **Reddit** [38]: Sequences of posts to sub-reddits that users make on the social website Reddit. Each sequence is a user, and the sub-reddits to which the user posts is considered as the mark.
- **Stack Overflow** [16]. Sequences of badges that users receive over time on the website Stack Overflow. A sequence is a specific user, and the type of badge received is the mark.

[4]https://pytorch.org/

We employ the pre-processed version of these datasets as described in [4] which can be openly accessed at this url: `https://www.dropbox.com/sh/maq7nju7v5020kp/AAAFBvzxeNqySRsAm-zgU7s3a/processed/data?dl=0&subfolder_nav_tracking=1` (MIT License). Specifically, each dataset is filtered to contain the 50 most represented marks, and all arrival-times are rescaled in the interval [0,10] to avoid numerical instabilities. To save computational time, the number of sequences in Reddit is reduced by 50%. Each dataset is randomly partitioned into 5 train/validation/test splits (60%/20%/20%). The summary statistics for each (filtered) dataset is reported in Table 3.

Table 3: Datasets statistics

	#Seq.	#Events	Mean Length	Max Length	Min Length	#Marks
LastFM	856	193441	226.0	6396	2	50
MOOC	7047	351160	49.8	416	2	50
Github	173	20656	119.4	4698	3	8
Reddit	4278	238734	55.8	941	2	50
Stack Overflow	7959	569688	71.6	735	40	22

A.4 Computational Time

We report in Table (4) the average execution time (in seconds) for a single forward and backward pass on all training sequences of all datasets. The results are averaged over 50 epochs. We notice that the computation of two separate embeddings $\boldsymbol{h}^t$ and $\boldsymbol{h}^m$ for THP**++**, SAHP**++** and FNN**++** inevitably leads to an increase in execution time, which appears more pronounced for larger datasets such as Reddit and Stack Overflow. However, the increased computational complexity is generally offset by improved model performance, as detailed in Sections 3 and A.7.

Table 4: Average execution time (in seconds) for a single forward and backward pass on all training sequences of all datasets. Results are averaged over 50 epochs.

	FNN		THP		SAHP	
	Base	**++**	Base	**++**	Base	**++**
LastFM	4.29	4.74	2.96	3.62	4.82	4.48
MOOC	6.36	8.07	5.19	5.76	8.08	7.74
Github	0.76	0.8	0.38	0.58	0.51	0.69
Reddit	19.13	20.32	9.75	13.5	13.63	15.9
Stack O.	32.83	33.95	16.32	23.1	21.11	26.45

A.5 Original model formulations

As basis for comparison, we provide in this section the original formulations of the neural MTPP models considered in this study.

SAHP. The marked intensity functions are given by

$$\lambda_k^*(t;\boldsymbol{\theta}) = \sigma_{S,k}\left(\boldsymbol{\mu}_k - (\boldsymbol{\eta}_k - \boldsymbol{\mu}_k)\text{exp}(-\boldsymbol{\gamma}_k(t - t_{i-1}))\right)_k, \tag{9}$$

where $\boldsymbol{\mu}_k = \sigma_G\left(\mathbf{W}_\mu \boldsymbol{h}\right)$, $\boldsymbol{\eta}_k = \sigma_G\left(\mathbf{W}_\eta \boldsymbol{h}\right)$, and $\boldsymbol{\gamma}_k = \sigma_{S,k}\left(\mathbf{W}_\gamma \boldsymbol{h}_i\right)$ with σ_G and $\sigma_{S,k}$ the GeLU [31] and mark-wise softplus activation functions, respectively, and $\mathbf{W}_\mu, \mathbf{W}_\eta, \mathbf{W}_\gamma \in \mathbb{R}^{K \times d_h}$.

THP. The marked intensity functions are given by

$$\lambda_k^*(t;\boldsymbol{\theta}) = \sigma_{S,k}\left(\boldsymbol{w}_t \frac{t - t_{i-1}}{t_{i-1}} + \mathbf{W}_h \boldsymbol{h} + \boldsymbol{b}\right)_k, \tag{10}$$

where $\boldsymbol{w}_t \in \mathbb{R}_+^K$, $\mathbf{W}_k \in \mathbb{R}^{K \times d_h}$, and $\boldsymbol{b} \in \mathbb{R}^K$.

FNN. The marked compensators are defined as

$$\Lambda_k^*(t;\boldsymbol{\theta}) = G_k^*(t;\boldsymbol{\theta}) - G_k^*(t_{i-1};\boldsymbol{\theta}), \tag{11}$$

$$G_k^*(t;\boldsymbol{\theta}) = \sigma_{S,k}\big(\mathbf{W}_2(\sigma_{GS,k}\big(\boldsymbol{w}_t(t-t_i) + \mathbf{W}_h\boldsymbol{h} + \boldsymbol{b}_1\big) + \boldsymbol{b}_2\big)_k, \tag{12}$$

where $\mathbf{W}_2 \in \mathbb{R}_+^{K\times d_1}$, $\boldsymbol{w}_t, \boldsymbol{b}_1 \in \mathbb{R}^{d_1}$, $\mathbf{W}_h \in \mathbb{R}^{d_1\times d_h}$, $\boldsymbol{b}_2 \in \mathbb{R}^K$, and $\sigma_{GS,k}$ is the mark-wise Gumbel-softplus activation function [17].

A.6 Alternative Scoring Rules

The NLL in (1) has been largely adopted as the default scoring rule for learning MTPP models [61]. However, our framework can be extended even further by using alternative scoring rules other than NLL for assessing the mark and time prediction tasks. Let S^f, S^m and S^w be (strictly) consistent scoring rules for $f^*(t;\boldsymbol{\theta}_\lambda) = \lambda^*(t;\boldsymbol{\theta}_\lambda)\exp\Big(-\sum_{k=1}^K \Lambda_k^*(t;\boldsymbol{\theta}_\lambda)\Big)$, $p^*(k|\tau;\boldsymbol{\theta}_p)$ and $1 - F^*(T;\boldsymbol{\theta}_\lambda) = \int_{t_n}^T \sum_{k=1}^K \lambda_k^*(s;\boldsymbol{\theta}_\lambda)ds$, respectively. Given a sequence $\mathcal{S}$ of n events (i.e. $L = 1$), the scoring rule

$$\mathcal{L}(\boldsymbol{\theta};\mathcal{S}) = \underbrace{\sum_{i=1}^n \big[S^f(f^*(t_i;\boldsymbol{\theta}_\lambda))\big] + S^w\left(1 - F^*(T;\boldsymbol{\theta}_\lambda)\right)}_{\mathcal{L}_T(\boldsymbol{\theta}_\lambda,\mathcal{S}_{\text{train}})} + \underbrace{\sum_{i=1}^n S^p(p^*(k_i|t_i;\boldsymbol{\theta}_p))}_{\mathcal{L}_M(\boldsymbol{\theta}_p,\mathcal{S}_{\text{train}})}, \tag{13}$$

is (strictly) consistent for the conditional joint density $f^*(t,k;\boldsymbol{\theta}_\lambda,\boldsymbol{\theta}_p)$ restricted to the interval $[0,T]$ [6]. Using the LogScore for S^f, S^m and S^w in (13) reduces to the NLL in (1). One can also use other choices tailored to the specific task. For instance, one can choose to use the continuous ranked probability score (CRPS) [23] for S^f to evaluate the predictive distribution of inter-arrival times [1]. Similarly, the Brier score [7] can be used for S^p to evaluate the predictive distribution of marks. Contrary to the *local* property of the LogScore, both the CRPS and the Brier score are *sensitive to distance*, in the sense that they reward predictive distributions that assign probability mass close to the observed realization. Nonetheless, the choice between local and non-local proper scoring rules has been generally subjective in the literature. While exploring alternative scoring rules to train neural MTPP models is an exciting research direction, we leave it as future work and train the models exclusively on the NLL in (1).

A.7 Additional Results

Distributions of conflicting gradients during training of the base models. Figures 5 and 6 show the distribution of $\cos\phi_{TM}$ during training for THP, SAHP, and FNN at the encoder (ENC) and decoder (DEC) heads on all datasets . The distribution is obtained as follows: each model is trained to minimize the NLL defined in (1) using the Adam optimizer [35] with $\alpha = 10^{-3}$. Consider that $\boldsymbol{\theta} = \{\boldsymbol{\theta}^p\}_{p=1}^P$, where $\boldsymbol{\theta}^p$ denote the learnable parameters of the p^{th} layer of the model, e.g. the weights of a feed-forward network. Given a batch of training sequences, we first evaluate $\mathcal{L}_T(\boldsymbol{\theta})$ and $\mathcal{L}_M(\boldsymbol{\theta})$ and compute the gradients $\boldsymbol{g}_T^p = \nabla_{\boldsymbol{\theta}^p}\mathcal{L}_T(\boldsymbol{\theta})$ and $\boldsymbol{g}_M^p = \nabla_{\boldsymbol{\theta}^p}\mathcal{L}_M(\boldsymbol{\theta})$ for all $\boldsymbol{\theta}^p \in \boldsymbol{\theta}$. We finally evaluate the values of $\cos\phi_{TM}$ according to definition (1) for all $\boldsymbol{\theta}^p$, and pool them over all training iterations. As discussed in the main text, THP, SAHP and FNN frequently exhibit (severe) conflicting gradients during training at both encoder and decoder heads, which impairs their performance on the time and mark prediction tasks.

Additional evaluation metrics. Besides the $\mathcal{L}_T$ metric reported in the main text to evaluate the time prediction task, we quantify the (unconditional) probabilistic calibration of the fitted models by computing the Probabilistic Calibration Error (PCE) [15]. Similarly, for the mark prediction task, we also quantify the probabilistic calibration of the mark predictive distribution by computing the Expected Calibration Error (ECE) [51]. Moreover, we assess the probabilistic calibration of both predictive distributions of arrival times and marks through reliability diagrams [24, 37]. Finally, by predicting the next event's mark as

$$\tilde{k} = \operatorname*{argmax}_{k\in\mathbb{K}} p^*(k|t;\boldsymbol{\theta}_p), \tag{14}$$

we can assess the quality of the point predictions by means of various classification metrics. Specifically, we compute the accuracy and the Mean Reciprocal Rank (MRR) [12] of mark predictions. Lower PCE and ECE is better, while higher accuracy and MRR is better.

Tables 5 and 6 give the PCE, ECE, MRR, and accuracy results for THP, SAHP and FNN in the base and base**++** configurations across all datasets. The metrics are averaged over 3 splits, and the standard error is given in parenthesis. We observe general improvement with respect to the time and mark prediction tasks when moving from the base to the base**++** models. These results are consistent with our previous conclusions, i.e. by preventing conflicting gradients during training, our framework leads to improved predictive accuracy and more reliable uncertainty estimates.

Table 5: PCE and ECE results of the different setups across all datasets. The values are computed over 3 splits, and the standard error is reported in parenthesis. Best results are highlighted in bold.

	PCE				
	LastFM	MOOC	Github	Reddit	Stack O.
THP	0.28 (0.0)	0.37 (0.0)	**0.21 (0.01)**	0.12 (0.0)	0.01 (0.0)
THP++	0.28 (0.01)	0.37 (0.0)	0.22 (0.02)	**0.05 (0.0)**	0.01 (0.0)
SAHP	0.07 (0.01)	0.12 (0.0)	0.07 (0.01)	0.09 (0.01)	**0.01 (0.0)**
SAHP++	**0.04 (0.01)**	**0.03 (0.0)**	**0.04 (0.01)**	**0.01 (0.0)**	0.03 (0.0)
FNN	0.04 (0.0)	0.09 (0.0)	0.03 (0.01)	0.08 (0.0)	0.05 (0.0)
FNN++	**0.03 (0.0)**	**0.01 (0.0)**	**0.02 (0.0)**	**0.01 (0.0)**	**0.0 (0.0)**

	ECE				
	LastFM	MOOC	Github	Reddit	Stack O.
THP	0.29 (0.05)	0.07 (0.0)	0.1 (0.02)	**0.02 (0.0)**	0.12 (0.0)
THP++	**0.03 (0.0)**	**0.02 (0.0)**	**0.09 (0.02)**	0.03 (0.0)	**0.02 (0.0)**
SAHP	0.36 (0.01)	0.14 (0.0)	**0.09 (0.02)**	0.07 (0.01)	0.03 (0.0)
SAHP++	**0.03 (0.0)**	**0.03 (0.01)**	0.11 (0.04)	**0.02 (0.0)**	**0.02 (0.0)**
FNN	0.26 (0.07)	0.04 (0.0)	0.09 (0.02)	0.03 (0.01)	0.03 (0.0)
FNN++	**0.03 (0.01)**	**0.02 (0.0)**	**0.08 (0.02)**	**0.02 (0.0)**	0.03 (0.0)

Table 6: Accuracy and MRR results of the different setups across all datasets. The percentage of the majority mark is shown in parentheses below each dataset, reflecting its proportion of the total number of marked events. The values are computed over 3 splits, and the standard error is reported in parenthesis. Best results are highlighted in bold.

	Accuracy				
	LastFM (3.9%)	MOOC (5.5%)	Github (51.6%)	Reddit (17.7%)	Stack O. (43.3%)
THP	0.19 (0.0)	0.4 (0.0)	0.59 (0.04)	0.82 (0.0)	0.47 (0.0)
THP++	**0.26 (0.01)**	**0.56 (0.0)**	**0.67 (0.01)**	0.82 (0.0)	**0.49 (0.0)**
SAHP	0.05 (0.0)	0.36 (0.01)	0.61 (0.02)	0.71 (0.02)	0.48 (0.0)
SAHP++	**0.25 (0.0)**	**0.56 (0.0)**	**0.68 (0.01)**	**0.82 (0.0)**	0.48 (0.0)
FNN	0.14 (0.01)	0.51 (0.0)	0.67 (0.01)	0.81 (0.0)	0.48 (0.0)
FNN++	**0.25 (0.01)**	**0.55 (0.0)**	**0.68 (0.01)**	**0.82 (0.0)**	0.48 (0.0)

	MRR				
	LastFM (3.9%)	MOOC (5.5%)	Github (51.6%)	Reddit (17.7%)	Stack O. (43.3%)
THP	0.33 (0.0)	0.6 (0.0)	0.74 (0.02)	0.87 (0.0)	0.67 (0.0)
THP++	**0.39 (0.01)**	**0.7 (0.0)**	**0.79 (0.01)**	0.87 (0.0)	0.67 (0.0)
SAHP	0.15 (0.0)	0.5 (0.01)	0.75 (0.01)	0.77 (0.02)	0.67 (0.0)
SAHP++	**0.39 (0.0)**	**0.7 (0.0)**	**0.79 (0.01)**	**0.87 (0.0)**	0.67 (0.0)
FNN	0.28 (0.01)	0.67 (0.0)	0.79 (0.01)	0.86 (0.0)	0.66 (0.0)
FNN++	**0.39 (0.01)**	**0.7 (0.0)**	**0.8 (0.01)**	**0.87 (0.0)**	**0.67 (0.0)**

Reliability diagrams. Figures 7 and 8 present the reliability diagrams for the predictive distributions of arrival-times and marks for the base**++** models on all datasets. Regarding $p^*(k|t;\boldsymbol{\theta}_p)$, the diagrams show that the base**++** models are in general better calibrated than their base counterparts, as evidenced by the bin accuracies aligning more closely with the diagonal. This improvement, which aligns with the ECE results of Table 5, indicates that $p^*(k|t;\boldsymbol{\theta}_p)$ obtained in the base**++** setting more faithfully reflects the true underlying uncertainty of the model. Additionally, we note that improvements with respect to the probabilistic calibration of the predictive distribution of arrival times are in general less prevalent. This observation is in accordance to the PCE results of Table 5 and suggests that conflicting gradients during training predominantly affect the mark prediction task. Nonetheless, we observe substantial time calibration improvements for SAHP**++** and FNN**++** on MOOC and Reddit compared to their base parametrizations.

Conflicting gradients remain harmful as capacity increases. Figure 3 shows the evolution of the proportion of conflicting gradients CG for the base THP, SAHP and FNN on LastFM, along with the evolution of their test $\mathcal{L}_T$ and $\mathcal{L}_M$. For each capacity (25K, 50K, 75K and 100K parameters), we maintain the distribution of parameters between the encoder and decoder heads constant at 0.67/0.33. We note that increasing a model's capacity has a limited impact on CG as well as on model performance with respect to both $\mathcal{L}_T$ and $\mathcal{L}_M$.

Scaling the loss does not efficiently address conflicts. To better balance tasks during training, a natural approach would consist in scaling the contribution of $\mathcal{T}_T$ in (1) to reduce its impact on the overall loss, i.e.

$$\mathcal{L}(\theta;\mathcal{S}_{\text{train}}) = \frac{1}{s}\mathcal{L}_T(\theta;\mathcal{S}_{\text{train}}) + \mathcal{L}(\theta;\mathcal{S}_{\text{train}}), \tag{15}$$

where $s \geq 1$ is a scaling coefficient. To assess the effectiveness of this method, we train the base THP, SAHP and FNN models on the objective in (15) following the experimental setup detailed in Section A.2. For these models, we report in Figure (4) the evolution of the training CG along their **unscaled**

test $\mathcal{L}_T$ and $\mathcal{L}_M$. We observe that the occurrence of conflicting gradients is marginally impacted by larger values of s.

However, as scaling increases, optimization begins to favor $\mathcal{T}_M$, and improvements on $\mathcal{L}_M$ can be observed. Nevertheless, this comes at the cost of significant degradation with respect to $\mathcal{L}_T$, offsetting the gains on $\mathcal{T}_M$. Although a specific value of s could lead to a trade-off between tasks, models trained in our base++ setting generally show improved performance with respect to both tasks simultaneously.

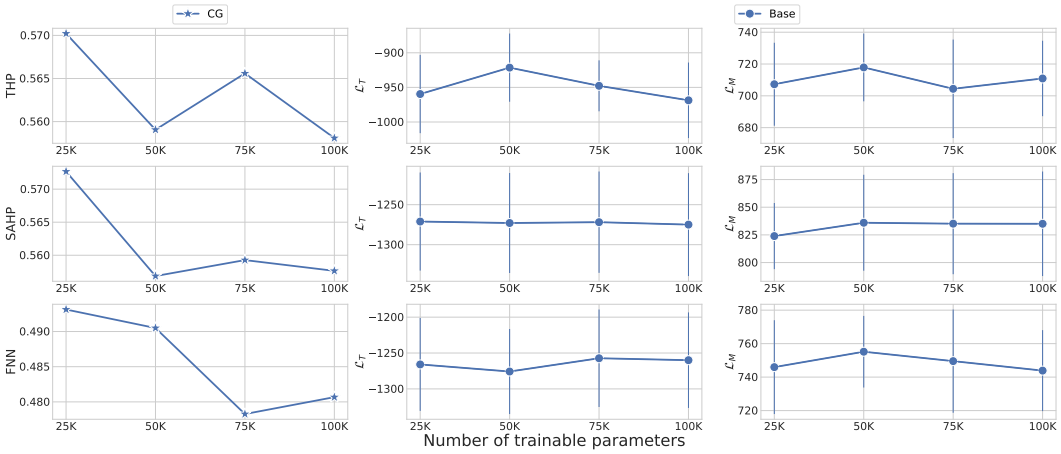


Figure 3: Evolution of CG, $\mathcal{L}_T$ and $\mathcal{L}_M$ with increasing model capacity for the base THP, SAHP and FNN models during training on LastFM.

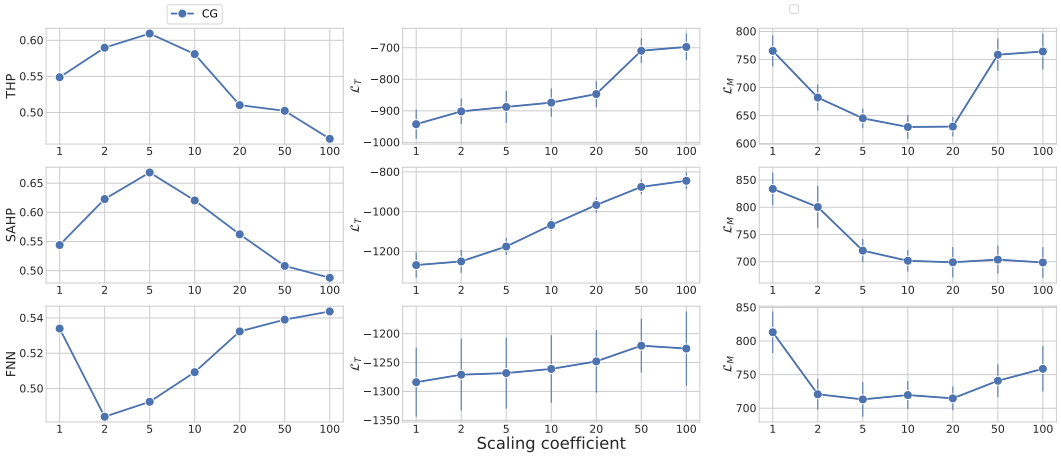


Figure 4: Evolution of CG, $\mathcal{L}_T$ and $\mathcal{L}_M$ with increasing value of scaling s in (15) for the base THP, SAHP and FNN models during training on LastFM.

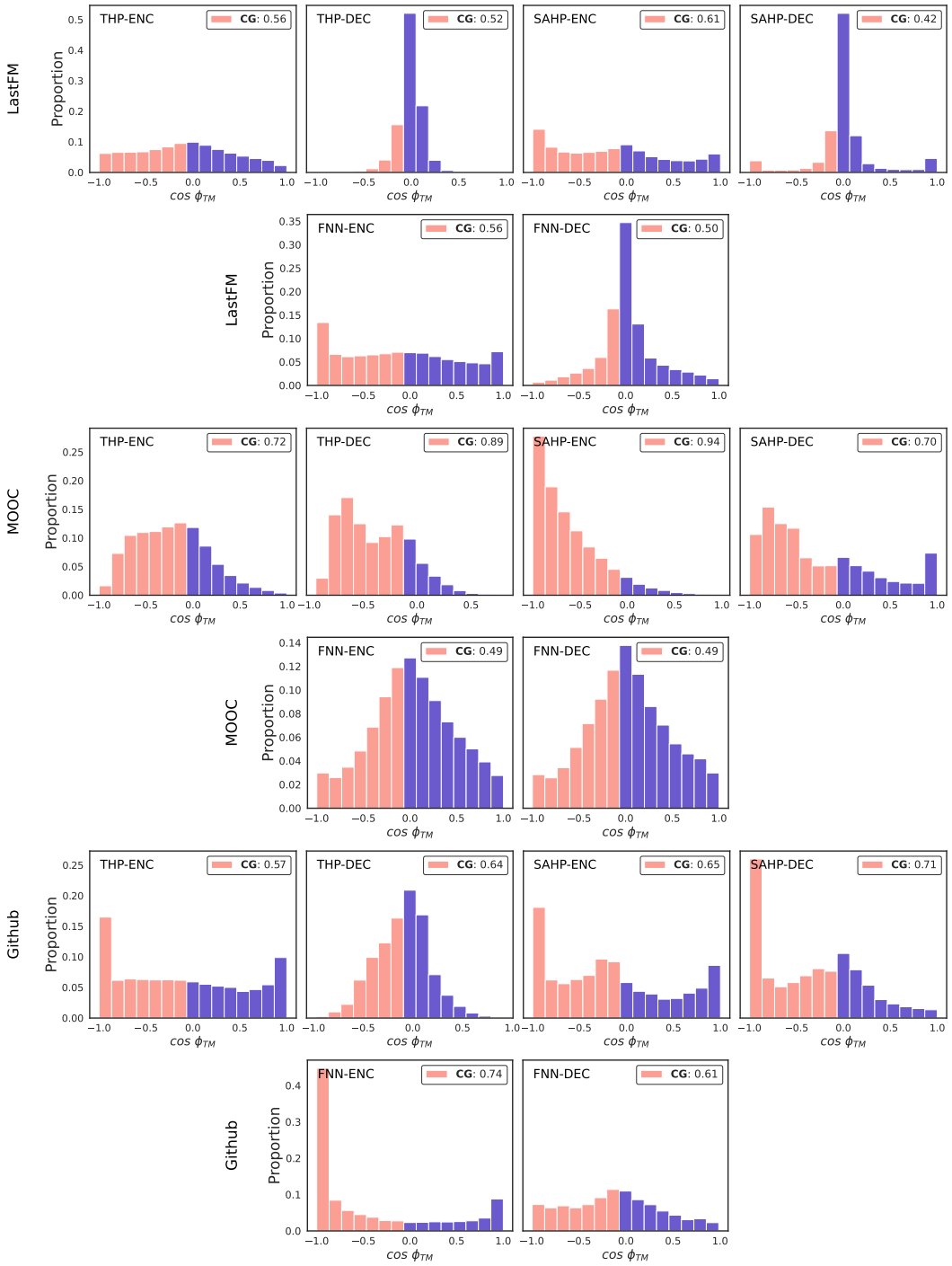


Figure 5: Distribution of cos ϕ_{TM} during training of THP, SAHP, and FNN on LastFM, MOOC and Github at the encoder (ENC) and decoder (DEC) heads. CG refers to the proportion of cos $\phi_{TM} < 0$ (red bars) observed during training. The distribution is obtained by pooling the values of ϕ_{TM} over 3 training runs.

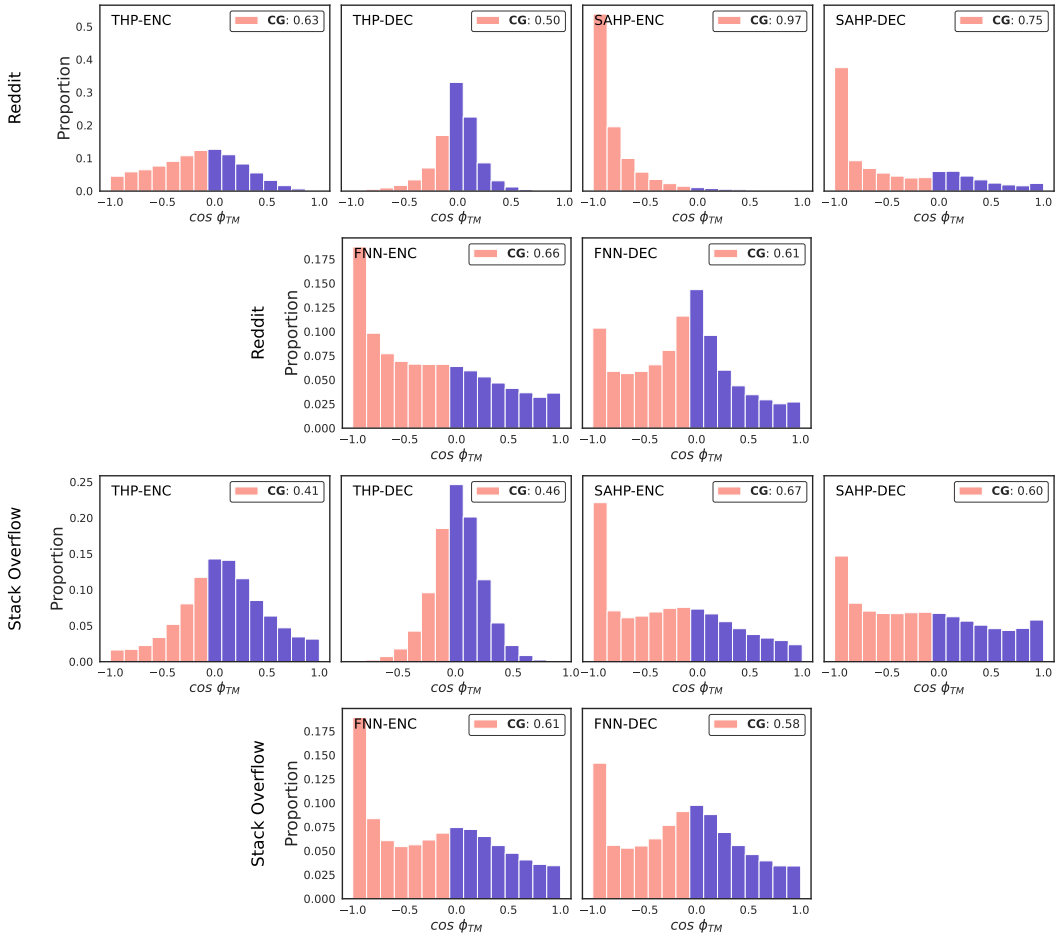


Figure 6: Distribution of cos ϕ_{TM} during training of THP, SAHP, and FNN on Reddit and Stack Overflow at the encoder (ENC) and decoder (DEC) heads. CG refers to the proportion of cos $\phi_{TM} < 0$ (red bars) observed during training. The distribution is obtained by pooling the values of ϕ_{TM} over 3 training runs.

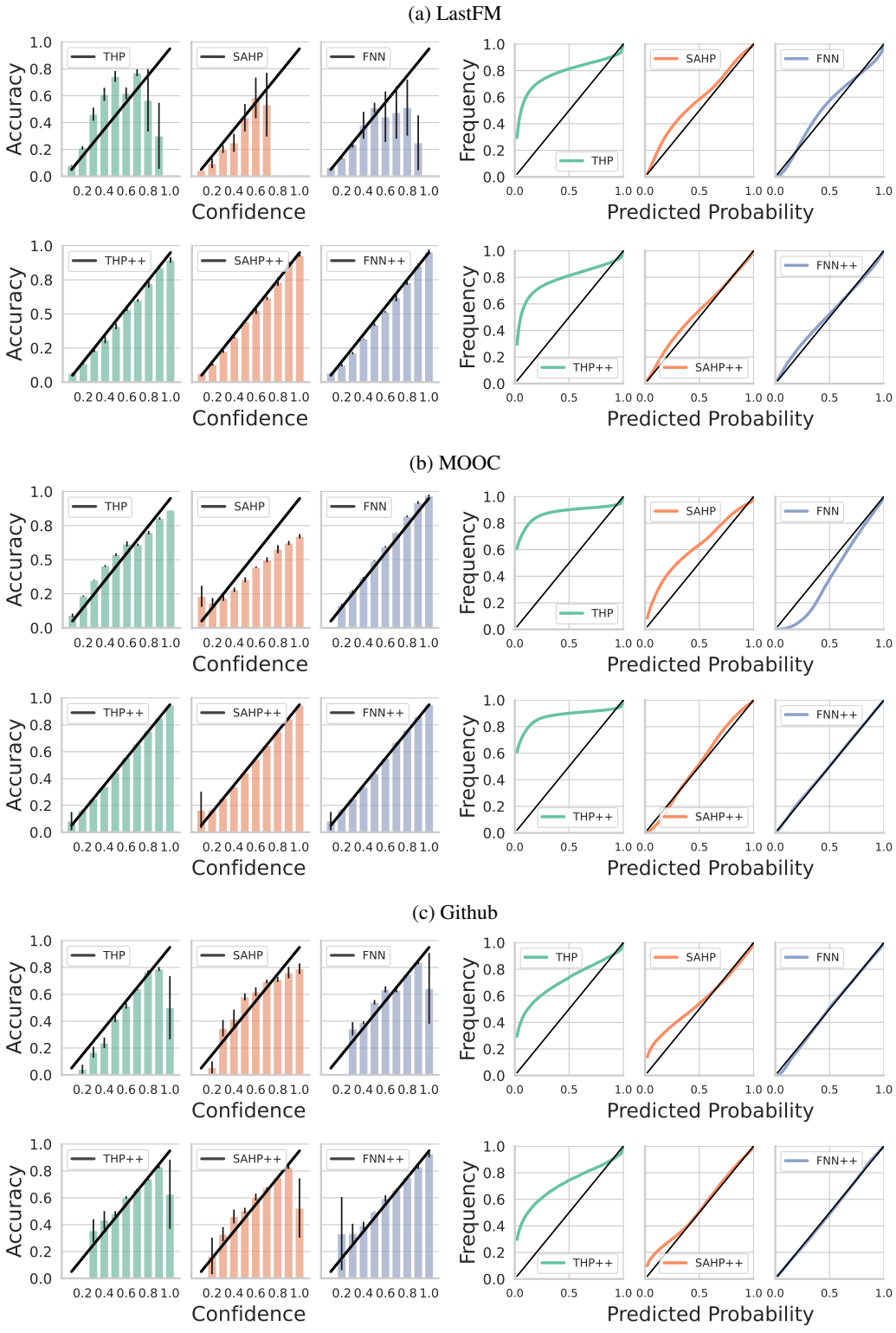
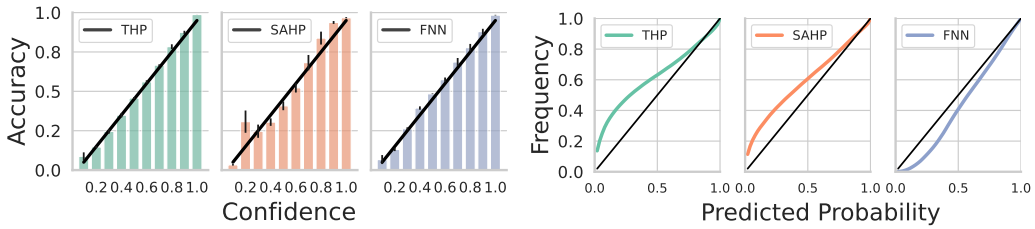


Figure 7: Reliability diagrams of the predictive $p^*(k|t;\boldsymbol{\theta}_p)$ (left) and $\lambda^*(t,\boldsymbol{\theta}_\lambda)/\Lambda^*(t;\boldsymbol{\theta}_\lambda)$ (right) on LastFM, MOOC, and Github. Frequency and Accuracy aligning with the black diagonal corresponds to perfect probabilistic and top label calibration, respectively. The results are averaged over 3 splits, and the error bars correspond to the standard error.

(a) Reddit



(b) Stack Overflow

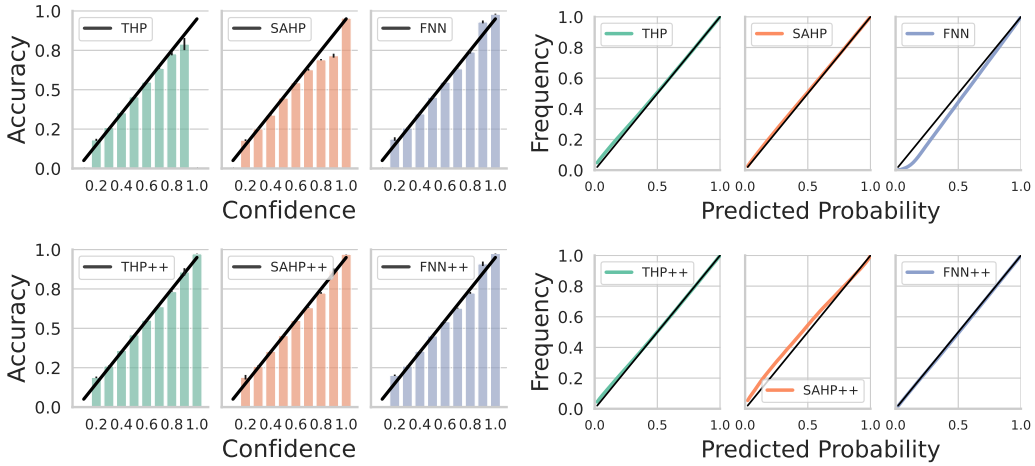


Figure 8: Reliability diagrams of the predictive $p^*(k|t;\boldsymbol{\theta}_p)$ (left) and $\lambda^*(t,\boldsymbol{\theta}_\lambda)/\Lambda^*(t;\boldsymbol{\theta}_\lambda)$ (right) on Reddit and Stack Overflow. Frequency and Accuracy aligning with the black diagonal corresponds to perfect probabilistic and top label calibration, respectively. The results are averaged over 3 splits, and the error bars correspond to the standard error.