

Supplementary Materials: Deep Continuous-Time State-Space Models for Marked Event Sequences

Table of Contents

- Appendix [A](#) Acronyms and Notation
- Appendix [B](#) Additional Details on Methods
- Appendix [C](#) Experimental Configurations and Datasets
- Appendix [D](#) Additional Experimental Results

A Acronyms and Notation

Table 3: Key notation used repeatedly across this paper.

Symbol	Space	Description
t	$\mathbb{R}_{\geq 0}$	Time
T	$\mathbb{R}_{\geq 0}$	Maximum time in a given sequence’s observation window
t_i	$\mathbb{R}_{\geq 0}$	i^{th} time
$t-$	$\mathbb{R}_{\geq 0}$	Subscript minus indicates left-limit
$t+$	$\mathbb{R}_{\geq 0}$	Subscript plus indicates right-limit
k	$\mathcal{M} = \{1, \dots, K\}$	Event mark
\mathcal{H}	$\mathcal{M}^N \times \mathbb{R}_{\geq 0}^N$	Event history for N events
\mathbf{N}_t	$\mathbb{Z}_{\geq 0}^K$	Counting process for K marks at time t
λ_t^k	$\mathbb{R}_{\geq 0}$	Intensity of k^{th} mark type at time t
$\boldsymbol{\lambda}_t$	$\mathbb{R}_{\geq 0}^K$	Vector of K mark intensities at time t
λ_t	$\mathbb{R}_{\geq 0}$	Ground/total intensity (sum of mark-specific intensities)
$\mathcal{L}(\cdot)$	\mathbb{R}	Log-likelihood of the argument under the model
ν^k	$\mathbb{R}_{\geq 0}$	Background intensity for the k^{th} mark
$\boldsymbol{\alpha}$	$\mathbb{R}_{\geq 0}^{K \times K}$	(For LHP) Matrix of intensity impulses from each type of mark
$\boldsymbol{\beta}$	$\mathbb{R}_{\geq 0}^{K \times K}$	(For LHP) Dynamics matrix of intensity evolution
R	\mathbb{Z}^+	Mark embedding rank
P	\mathbb{Z}^+	LLH/SSM hidden dimension
\mathbf{x}_t	\mathbb{R}^P	LLH/SSM hidden state at time t
\mathbf{x}_0	\mathbb{R}^P	Learned LLH/SSM initial hidden state
H	\mathbb{N}	LLH/SSM output dimension
\mathbf{y}_t	\mathbb{R}^H	LLH/SSM output at time t
\mathbf{u}_t	\mathbb{R}^H	LLH/SSM input at time t
\mathbf{A}	$\mathbb{R}^{P \times P}$	LLH/SSM transition matrix
\mathbf{B}	$\mathbb{R}^{P \times H}$	LLH/SSM input matrix
\mathbf{C}	$\mathbb{R}^{H \times P}$	LLH/SSM output matrix
\mathbf{D}	$\mathbb{R}^{H \times H}$	LLH/SSM passthrough matrix
\mathbf{E}	$\mathbb{R}^{P \times R}$	LLH mark embedding matrix ($P \times R$ in low-rank factorization)
L	\mathbb{Z}^+	Number of linear recurrences in a S2P2 model; model “depth”
$\boldsymbol{\alpha}$	$\mathbb{R}^{R \times K}$	(For S2P2) Mark impulses ($R \times K$ in low-rank factorization)
\sim	N/A	Tilde (e.g., \mathbf{B}) denotes variable is in the diagonalized eigenbasis
Λ	$\mathbb{C}^{P \times P}$	Matrix of eigenvalues of \mathbf{A} ; diagonalized dynamics matrix
$\bar{\Lambda}$	$\mathbb{C}^{P \times P}$	Discretized diagonal dynamics matrix
(l)	N/A	Superscript index in parenthesis indicates layer (i.e., \mathbf{x} for layer l)

Table 4: Key acronyms used throughout this paper.

Acronym	Page number	Definition
CNN	7	Convolutional neural network
LHP	2	Linear Hawkes process
LLH	2	Latent linear Hawkes
MTTP	1	Marked temporal point process
RNN	1	Recurrent neural network
SSM	1	(Deep) State-space model
TPP	7	Temporal point process
ZOH	5	Zero-order hold
RMTPP	2	Recurrent marked temporal point process [Du et al., 2016]
NHP	2	Neural Hawkes process [Mei and Eisner, 2017]
SAHP	2	Self-attentive Hawkes process [Zhang et al., 2020]
THP	2	Transformer Hawkes process [Zuo et al., 2020]
AttNHP	2	Attentive neural Hawkes process [Yang et al., 2022]
IFTTP	2	Intensity-free temporal point process [Shchur et al., 2020a]
MHP	2	Mamba Hawkes process [Gao et al., 2024]
S2P2	1	State-space point process (ours)

B Additional Details on Methods

B.1 State-Space Point Process Algorithms

Algorithm 1 State-Space Point Process: Get Right State Limits

Input: S2P2 layer parameters $\Theta = \{\mathbf{\Lambda}^{(l)}, \tilde{\mathbf{B}}^{(l)}, \tilde{\mathbf{C}}^{(l)}, \mathbf{D}^{(l)}, \tilde{\mathbf{E}}^{(l)}, \tilde{\mathbf{x}}_0^{(l)}\}_{l=1}^L$, event intervals $\Delta t_{1:N}$, nonlinearity σ , shared mark embeddings $\alpha_{1:N}$.

Output: Right state limits $\mathbf{x}_{t_{1:N}}^{(1:L)}$

```

1:  $\mathbf{u}_{t_{1:N}-} = \mathbf{0}$  ▷ Left input limits
2: for  $l$  in  $1 : L$  do
3:    $\bar{\mathbf{\Lambda}}_{1:N}^{(l)} = \text{Discretize}(\mathbf{\Lambda}^{(l)}, \Delta t_{1:N})$  ▷ Zero-order hold, see Eq. (19)
4:    $\tilde{\mathbf{x}}_{t_{1:N}}^{(l)} = \text{ParallelScan}(\bar{\mathbf{\Lambda}}_{1:N}^{(l)}, (\bar{\mathbf{\Lambda}}_{1:N}^{(l)} - \mathbf{I})\tilde{\mathbf{B}}^{(l)}\mathbf{u}_{t_{1:N}-} + \tilde{\mathbf{E}}^{(l)}\alpha_{1:N})$  ▷ Compute right  $x$  limits
5:    $\tilde{\mathbf{x}}_{t_{1:N}-}^{(l)} = \tilde{\mathbf{x}}_{t_{1:N}}^{(l)} - \tilde{\mathbf{E}}^{(l)}\alpha_{1:N}$  ▷ Compute left  $x$  limits
6:    $\mathbf{u}_{t_{1:N}-} = \text{LayerNorm}(\sigma(\tilde{\mathbf{C}}^{(l)}\tilde{\mathbf{x}}_{t_{1:N}-} + \mathbf{D}^{(l)}\mathbf{u}_{t_{1:N}-}) + \mathbf{u}_{t_{1:N}-})$  ▷ Compute next layer's left  $u$  limits
7: end for
8: return  $\mathbf{x}_{t_{1:N}}^{(1:L)}$ 

```

Algorithm 2 State-Space Point Process: Get Intensity From Right Limit

Input: S2P2 layer parameters $\Theta = \{\mathbf{\Lambda}^{(l)}, \tilde{\mathbf{B}}^{(l)}, \tilde{\mathbf{C}}^{(l)}, \mathbf{D}^{(l)}, \tilde{\mathbf{E}}^{(l)}, \tilde{\mathbf{x}}_0^{(l)}\}_{l=1}^L$, Previous state right limits $\mathbf{x}_t^{(1:L)}$, Integration period δt , nonlinearity σ , Intensity function IntensityFn.

Output: Intensity left limit $\lambda_{t+\delta t-}$

```

1:  $\mathbf{u}_{t+\delta t-} = \mathbf{0}$  ▷ Left input limit
2: for  $l$  in  $1 : L$  do
3:    $\bar{\mathbf{\Lambda}}^{(l)} = \text{Discretize}(\mathbf{\Lambda}^{(l)}, \delta t)$  ▷ Zero-order hold, see Eq. (19)
4:    $\tilde{\mathbf{x}}_{t+\delta t-}^{(l)} = \bar{\mathbf{\Lambda}}^{(l)}\mathbf{x}_t^{(l)} + (\bar{\mathbf{\Lambda}}^{(l)} - \mathbf{I})\tilde{\mathbf{B}}^{(l)}\mathbf{u}_{t+\delta t-}$  ▷ Evolve state
5:    $\mathbf{u}_{t+\delta t-} = \text{LayerNorm}(\sigma(\tilde{\mathbf{C}}^{(l)}\tilde{\mathbf{x}}_{t+\delta t-}^{(l)} + \mathbf{D}^{(l)}\mathbf{u}_{t+\delta t-}) + \mathbf{u}_{t+\delta t-})$  ▷ Compute event left  $u$  limits
6: end for
7:  $\lambda_{t+\delta t-} = \text{IntensityFn}(\mathbf{u}_{t+\delta t-})$  ▷ Rectify intensity, see Section 3.4
8: return  $\lambda_{t+\delta t}$ 

```

Algorithm 3 State-Space Point Process: Compute Log-Likelihood

Input: S2P2 layer parameters $\Theta = \{\mathbf{\Lambda}^{(l)}, \tilde{\mathbf{B}}^{(l)}, \tilde{\mathbf{C}}^{(l)}, \mathbf{D}^{(l)}, \tilde{\mathbf{E}}^{(l)}, \tilde{\mathbf{x}}_0^{(l)}\}_{l=1}^L$, Event times $t_{1:N}$, mark types $k_{1:N}$, nonlinearity σ , shared mark embedding function EmbedMarks, number of integration points per event M , Intensity function IntensityFn.

Output: Log-likelihood \mathcal{L}

```

1:  $\alpha_{1:N} = \text{EmbedMarks}(k_{1:N})$  ▷ Shared embeddings
2:  $t_0 := 0$ 
3:  $\Delta t_{1:N} = t_{1:N} - t_{0:N-1}$ 
4:  $s_{1:N,1:M} \sim \mathcal{U}(0, \Delta t_{1:N})$  ▷ Sample  $M$  integration points per interval (non-inclusive)
5:  $\tilde{\mathbf{x}}_{t_{1:N}}^{(1:L)} = \text{GetRightStateLimits}(\Theta, \Delta t_{1:N}, \sigma, \alpha_{1:N})$  ▷ Algorithm 1  $\mathcal{O}(\log N)$  parallel time
6: for  $n$  in  $1 : N$  do ▷ This is embarrassingly parallelizable with vmap,  $\mathcal{O}(1)$  parallel time
7:    $\lambda_{t_n} = \text{GetIntensityFromRightLimit}(\Theta, \tilde{\mathbf{x}}_{t_n}^{(1:L)}, \Delta t_n, \sigma, \text{IntensityFn})$  ▷ Algorithm 2  $\mathcal{O}(1)$  parallel time
8:   for  $m$  in  $1 : M$  do ▷ This is embarrassingly parallelizable with vmap,  $\mathcal{O}(1)$  parallel time
9:      $\lambda_{s_{n,m}} = \text{GetIntensityFromRightLimit}(\Theta, \tilde{\mathbf{x}}_{t_n}^{(1:L)}, s_{n,m}, \sigma, \text{IntensityFn})$  ▷ Algorithm 2  $\mathcal{O}(1)$  parallel time
10:   end for
11: end for
12:  $\mathcal{L} = \sum_{n=1}^N \log \lambda_{t_n}^{k_n} + \sum_{n=1}^N \frac{\Delta t_n}{M} \sum_{m=1}^M \sum_{k=1}^K \lambda_{s_{n,m}}^{k_n}$  ▷ Eq. (2) with Monte-Carlo approximation of integral
13: return  $\mathcal{L}$ 

```

B.2 Discretization and Zero Order Hold

The linear recurrence is defined in continuous-time. This mirrors the (M)TPP setting, where event times are not on a fixed intervals. We use the zero-order hold (ZOH) discretization method, to convert the continuous-time linear recurrence into a sequence of closed-form updates, given the integration times, that can also be efficiently computed. We refer the reader to [Iserles \[2009\]](#) for a comprehensive introduction to the ZOH transform.

The main assumption of the ZOH discretization is that the input signal is held constant over the time period being integrated. Under this assumption, it is possible to solve for the dynamics and input matrices that yield the correct state at the end of the integration period. For the LLH dynamics in Eq. (8), when no events occur in (t, t') , this becomes

$$\mathbf{x}_{t'-} = \int_t^{t'} \mathbf{A}\mathbf{x}_t + \mathbf{A}\mathbf{B}\mathbf{u}_t dt = \bar{\mathbf{A}}\mathbf{x}_t + \bar{\mathbf{A}}\mathbf{B}\mathbf{u}_t \quad \text{assuming} \quad d\mathbf{u}_t = \mathbf{0} \in [t, t'], \quad (15)$$

where the resulting discretized matrices are

$$\bar{\mathbf{A}} = e^{\mathbf{A}\Delta t}, \quad \bar{\mathbf{A}}\mathbf{B} = \mathbf{A}^{-1}(e^{\mathbf{A}\Delta t} - \mathbf{I})\mathbf{A}\mathbf{B}, \quad \text{where} \quad \Delta t = t' - t. \quad (16)$$

The ZOH does not affect the output or passthrough matrices \mathbf{C} and \mathbf{D} . To compute the matrices $\bar{\mathbf{A}}$ and $\bar{\mathbf{A}}\mathbf{B}$ however requires computing a matrix exponential and a matrix inverse. [Smith et al. \[2022\]](#) avoid this by diagonalizing the system (also avoiding a dense matrix-matrix multiplication in the parallel scan). The diagonalized dynamics and input matrices are denoted $\mathbf{\Lambda}$ (a diagonal matrix) and $\mathbf{\Lambda}\tilde{\mathbf{B}}$ respectively. In this case, Eq. (16) reduces to

$$\bar{\mathbf{A}} = e^{\mathbf{\Lambda}\Delta t}, \quad (17)$$

$$\bar{\mathbf{A}}\mathbf{B} = \mathbf{\Lambda}^{-1}(e^{\mathbf{\Lambda}\Delta t} - \mathbf{I})\mathbf{\Lambda}\tilde{\mathbf{B}} \quad (18)$$

$$= (e^{\mathbf{\Lambda}\Delta t} - \mathbf{I})\tilde{\mathbf{B}} \quad (\text{diagonal matrices commute}) \quad (19)$$

where $e^{\mathbf{\Lambda}\Delta t}$ is trivially computable as the exponential of the leading diagonal of $\mathbf{\Lambda}\Delta t$. These operations are embarrassingly parallelizable across the sequence length and state dimension given the desired evaluation times.

To contextualize, suppose an event occurs at time t , Eq. (19) allows us to exactly (under the constant-input assumption) efficiently evaluate the linear recurrence at subsequent times t' . We use this extensively in S2P2 to efficiently evaluate the recurrence (and hence the intensity) at the irregularly-spaced event times and times used to compute the integral term.

It should be noted the discretization was done to compute a left-limit $\mathbf{x}_{t'-}$ from a previous right-limit \mathbf{x}_t . Should an event not occur at t' , then the left- and right-limits agree and $\mathbf{x}_{t'-} = \mathbf{x}_{t'+} = \mathbf{x}_{t'}$. If an event does occur at time t' with mark k , then the left-limit $\mathbf{x}_{t'-}$ can be incremented by $\tilde{\mathbf{E}}\alpha_k$ to compute $\mathbf{x}_{t'+} = \mathbf{x}_{t'}$. This increment is exact and leverages no discretization assumption.

B.3 Interpretation for Input-Dependent Dynamics

Consider the input-dependent recurrence for an LLH layer, as defined in Eq. (13):

$$d\tilde{\mathbf{x}}_t := \mathbf{\Lambda}_i \tilde{\mathbf{x}}_{t-} dt + \mathbf{\Lambda}_i \tilde{\mathbf{B}} \mathbf{u}_{t-} dt + \tilde{\mathbf{E}} \alpha d\mathbf{N}_t \quad (20)$$

for $t \in (t_i, t_{i+1}]$ where $\mathbf{\Lambda}_i := \text{diag}(\Delta_i)\mathbf{\Lambda}$ with the input-dependent factor defined as $\Delta_i := \text{softplus}(\mathbf{W}'\mathbf{u}_{t_i} + \mathbf{b}') \in \mathbb{R}_{>0}^P$. This factor can be thought of as the input-dependent relative-time scale for the dynamics. To see this, we first note that for vectors $\mathbf{p}, \mathbf{q} \in \mathbb{R}^d$, the following holds true: $\text{diag}(\mathbf{p})\mathbf{q} = \mathbf{p} \odot \mathbf{q} = \mathbf{q} \odot \mathbf{p}$ where \odot is the Hadamard or element-wise product. It then follows that

$$d\tilde{\mathbf{x}}_t := \mathbf{\Lambda}_i \tilde{\mathbf{x}}_{t-} dt + \mathbf{\Lambda}_i \tilde{\mathbf{B}} \mathbf{u}_{t-} dt + \tilde{\mathbf{E}} \alpha d\mathbf{N}_t \quad (21)$$

$$= \mathbf{\Lambda}_i (\tilde{\mathbf{x}}_{t-} + \tilde{\mathbf{B}} \mathbf{u}_{t-}) dt + \tilde{\mathbf{E}} \alpha d\mathbf{N}_t \quad (22)$$

$$= \text{diag}(\Delta_i) \mathbf{\Lambda} (\tilde{\mathbf{x}}_{t-} + \tilde{\mathbf{B}} \mathbf{u}_{t-}) dt + \tilde{\mathbf{E}} \alpha d\mathbf{N}_t \quad (23)$$

$$= [\mathbf{\Lambda} (\tilde{\mathbf{x}}_{t-} + \tilde{\mathbf{B}} \mathbf{u}_{t-})] \odot (\Delta_i dt) + \tilde{\mathbf{E}} \alpha d\mathbf{N}_t. \quad (24)$$

As shown, the positive vector Δ_i can be thought of as changing the relative time-scale for each channel in the hidden state $\tilde{\mathbf{x}}$. Large values of Δ_i will act as if time is passing quickly, encouraging the state to converge to the steady-state sooner. Conversely, smaller values will make time pass more slowly causing the model to retain the influence that prior events have on future ones (for that specific channel in $\tilde{\mathbf{x}}$ at least).

B.4 Forwards and Backwards Zero Order Hold Discretization

In Section 3.2 we highlighted that the ZOH discretization is exact when \mathbf{u}_t is held constant over the integration window. This raises a unique design question for S2P2: what constant value should \mathbf{u}_t take on when evolving \mathbf{x} from time t to t' ? For the first layer of the model, the input is zero by construction, so there is no choice to be made—in fact, since \mathbf{u} is constant for the first layer the updates are exact. However, the input is non-zero at deeper layers, and, crucially, varies over the integration period.

We must therefore decide how to select a \mathbf{u} value over the integration period. This should be a value in (or function of) $\{\mathbf{u}_s \mid s \in [t, t']\}$. Note this is because the value at t' , $\mathbf{u}_{t'}$, *cannot* be incorporated as this would cause a data leakage in our model; while values prior to t would discard the most recent event occurrence. For this work, we explore two natural choices: (i) the input value at the beginning of the interval, \mathbf{u}_t , and (ii) the left-limit at the end of the interval, $\mathbf{u}_{t'-}$. Note the end of the interval need not align with an event (crucial for when computing intermediate intensity values). We refer to these options as *forwards* and *backwards* ZOH, respectively. We illustrate the backwards variant in Fig. 2, where in the leftmost panel, we use the \mathbf{u}_{t-} values at each layer to calculate \mathbf{x}_{t-} and \mathbf{x}_t , as opposed to \mathbf{u}_{t_i} for $t_i < t < t_{i+1}$. All experiments in the main paper utilize backwards ZOH.

It is not obvious *a priori* which one of these modes is more performant. We therefore conducted an ablation experiment in Table 14. We see that there is little difference between the two methods. We also note that models are learned *through* this discretization, and so this decision does not mean that a model is “incorrectly discretized” one way or the other, but instead they define subtlety different families of models. Theoretical and empirical investigation of the interpretations of this choice is an interesting area of investigation going forwards, extending the ablations we present in Table 14.

B.5 Theoretical Complexity

We include in Table 5 a brief summary of the theoretical complexity of each of the methods we consider, broadly grouped by their architectures. We analyze complexity by the work, memory complexity and theoretical best parallel application time of the forward pass (used when conditioning on a sequence, the left-hand term of Eq. (2)) and evaluating the integral term in Eq. (2) *given that the forward pass has been completed* (as this is either required by the method, and is nearly always evaluated in conjunction with the forward pass). We then state the limiting best-case theoretical parallelism of the two components.

The reasoning behind the calculated values are as follows:

- The forward pass of RMTTP, NHP and IFTTP use non-linear RNNs, and hence incur memory and work that is linear in the sequence length, and cannot be parallelized. MHP uses an RNN, but that is logarithmically parallelizable. These models re-use the computed hidden states to compute the integral term, and hence, while they incur work and memory that scales in the sequence length and number of events, this work can be perfectly parallelized. This results in a best-case parallelism of $\mathcal{O}(N)$ (dominated by the forward pass; $\mathcal{O}(\log N)$ for MHP).
- SAHP, THP and AttNHP all use self-attention, and hence have a work and memory that scales quadratically in the sequence length, although this work can be parallelized across the sequence length, resulting in logarithmic parallel depth. SAHP and THP re-use embeddings and a parametric decoder, and hence estimating the integral scales like the RNN, and hence the limiting parallelism is still the forward pass. AttNHP is slightly different in that it re-applies the whole independently attention mechanism for each integration point. However, this work is parallelizable and hence still reduces to a best-case depth of $\mathcal{O}(\log N)$.
- S2P2 is an RNN and hence has linear work and memory in the forward pass, but can be parallelized to a best-case depth of $\mathcal{O}(\log N)$ using the parallel scan. We then re-use the states computed in the forward pass for estimating the integral, which, as with the other RNN methods, is perfectly parallelizable, resulting in a theoretical parallel depth of $\mathcal{O}(\log N)$.

Note that these figures do not account for the number of layers required by each model, which must be evaluated in sequence.

Table 5: Comparison of methods based on memory and compute complexity. We see that our S2P2 matches the best performing baseline in all categories. N denotes to the sequence length, and M denotes to the number of Monte Carlo grid points per-event used in evaluating Eq. (2). As IFTPP is an intensity-free method, it does not need to estimate $\int \lambda_t dt$ as the other methods do.

Method	Forward Pass			Estimating $\int \lambda_t dt$			Overall
	Memory	Work	Theoretical Parallelism	Memory	Work	Theoretical Parallelism	Theoretical Parallelism
RMTTP	$\mathcal{O}(N)$	$\mathcal{O}(N)$	$\mathcal{O}(N)$	$\mathcal{O}(NM)$	$\mathcal{O}(NM)$	$\mathcal{O}(1)$	$\mathcal{O}(N)$
NHP	$\mathcal{O}(N)$	$\mathcal{O}(N)$	$\mathcal{O}(N)$	$\mathcal{O}(NM)$	$\mathcal{O}(NM)$	$\mathcal{O}(1)$	$\mathcal{O}(N)$
MHP	$\mathcal{O}(N)$	$\mathcal{O}(N)$	$\mathcal{O}(\log N)$	$\mathcal{O}(NM)$	$\mathcal{O}(NM)$	$\mathcal{O}(1)$	$\mathcal{O}(\log N)$
IFTTP	$\mathcal{O}(N)$	$\mathcal{O}(N)$	$\mathcal{O}(N)$	N/A	N/A	N/A	$\mathcal{O}(N)$
SAHP	$\mathcal{O}(N^2)$	$\mathcal{O}(N^2)$	$\mathcal{O}(\log N)$	$\mathcal{O}(NM)$	$\mathcal{O}(NM)$	$\mathcal{O}(1)$	$\mathcal{O}(\log N)$
THP	$\mathcal{O}(N^2)$	$\mathcal{O}(N^2)$	$\mathcal{O}(\log N)$	$\mathcal{O}(NM)$	$\mathcal{O}(NM)$	$\mathcal{O}(1)$	$\mathcal{O}(\log N)$
AttNHP	$\mathcal{O}(N^2)$	$\mathcal{O}(N^2)$	$\mathcal{O}(\log N)$	$\mathcal{O}(N^2 M)$	$\mathcal{O}(N^2 M)$	$\mathcal{O}(\log N)$	$\mathcal{O}(\log N)$
S2P2	$\mathcal{O}(N)$	$\mathcal{O}(N)$	$\mathcal{O}(\log N)$	$\mathcal{O}(NM)$	$\mathcal{O}(NM)$	$\mathcal{O}(1)$	$\mathcal{O}(\log N)$

To validate the scaling properties, we measure the wallclock time for a full forward pass and log-likelihood evaluation on random input sequences with lengths ranging from eight events to over half a million events. The architectures and mark spaces are the same as in the StackOverflow experiments (see Tables 6a and 7).

We note: our EasyTPP PyTorch S2P2 is written in pure PyTorch, and hence is not as optimized as other methods (compared to, for instance, IFTTP, which uses a GPU-optimized implementation of the GRU). We therefore include the runtimes of a standalone JAX S2P2 implementation, which allows for comparable levels of optimization through JIT compilation.

We observe the predicted scaling in practice. NHP scales linearly across all sequence lengths, and is far outpaced by all other methods. The THP scales well before reverting to superlinear scaling, and then runs out of memory. The IFTTP is very fast at shorter runtimes, but quickly reverts to linear scaling, due to its simple but highly optimized implementation and inherently sequential operation. Both S2P2 implementations scale linearly at long sequence lengths, but have near-constant runtime at shorter sequences. At shorter sequence lengths, the more optimized JAX implementation is faster than the unoptimized pure PyTorch implementation. While this indicates that there are additional opportunities to accelerate the PyTorch implementation further (e.g., exploiting kernel fusion or writing a lower-level Triton implementation), these results still confirm that our S2P2 can exploit parallel scans to scale to long sequences more effectively than other methods while retaining strong performance.

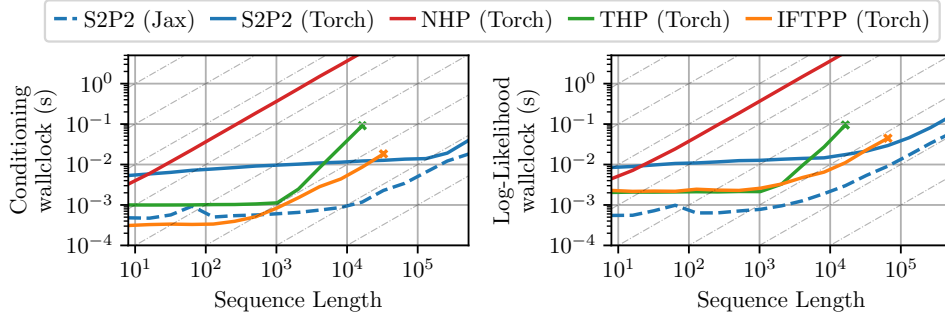


Figure 5: Median runtime of various models against increasing sequence lengths when conditioning on a sequence (Algorithm 1) and for likelihood evaluation (Algorithm 3) over 20 random seeds (variance negligible). S2P2 is faster across a wide range of sequence lengths. Crosses indicate where THP runs out of memory or IFTTP throws an error.

C Experimental Configurations and Datasets

C.1 Training Details & Hyperparameter Configurations

All baseline models used up-to-date PyTorch implementations, provided by the EasyTPP library [Xue et al., 2023] as of May 2025.

We apply a grid search for all models on all datasets for hyperparameter tuning. We use a default batch size of 256 for training. For models/datasets that require more memory (e.g., large mark space or long sequences), we reduce the batch size and keep them as consistent as possible among all the models on each dataset. We use the Adam stochastic gradient optimizer [Kingma and Ba, 2015], with a learning rate of 0.01 and a linear warm-up schedule over the first 1% iterations, followed by a cosine decay. Initial experiments showed this setting generally worked well across different models and datasets leads to convergence within 300 epochs. We also clip the gradient norm to have a max norm of 1 for training stability. We use Monte-Carlo samples to estimate the integral in log-likelihood, where we use 10 Monte-Carlo points per event during training.

On the five EasyTPP benchmark datasets and MIMIC-II that are smaller in their scales, we choose an extended grid based on the architecture reported in the EasyTPP paper. Specifically, we search over hidden states size $h = \{16, 32, 64, 128, 256\}$ for RMTTP, $h = \{32, 64, 128\}$ for NHP, and $h = \{16, 32, 64\}$ for IFTTP. For SAHP, THP, and AttNHP, we searched over all combinations of number of $L = \{1, 2, 3\}$, hidden state size $= \{16, 32, 64, 128\}$, and number of heads $= \{1, 2, 4\}$. For MHP, we followed their paper to fix $L = 4$, then we search over $h = \{4, 8, 16, 32, 64, 128, 256, 512\}$. Finally, for S2P2, we considered combinations for number of layers $= \{1, 2, 3, 4\}$, $p = \{16, 32, 64, 128\}$ and $h = \{16, 32, 64, 256\}$. We generally found a range of reasonable hyperparameters to yield similar performance on S2P2, while multiple layers were key for performance (intuitively deep stack allows nonlinear and complex dynamics), but there was not a critical dependence on depth.

We fixed the activation function as GeLU [Hendrycks and Gimpel, 2016] and apply post norm with layer norm [Ba, 2016]. We fix the dropout as 0.1 for S2P2 on the five core benchmark datasets, and add dropout $= \{0, 0.1\}$ to the grid search for the other three datasets. Due to the scale of Last.fm and EHRSHOT datasets, we perform a smaller search over architectures that roughly match the parameter counts for all models at three levels: 25k, 50k, 200k, and choose the model with the best validation results. AttNHP has expensive memory requirements that tends to have smaller batch sizes than other models. We were unable to train any AttNHP on EHRSHOT. The final model architectures used are reported in Tables 6a and 6b.

Table 6: Model architectures for the five EasyTPP benchmark datasets in Table 2.

(a) Model architectures for the five EasyTPP benchmark datasets in Table 2

Model	Amazon	Retweet	Taxi	Taobao	StackOverflow
RMTTP	$h = 128$	$h = 16$	$h = 128$	$h = 16$	$h = 256$
SAHP	$h = 32, l = 2, \text{heads} = 2$	$h = 32, l = 3, \text{heads} = 4$	$h = 16, l = 2, \text{heads} = 4$	$h = 32, l = 1, \text{heads} = 1$	$h = 64, l = 1, \text{heads} = 1$
THP	$h = 32, l = 2, \text{heads} = 4$	$h = 16, l = 3, \text{heads} = 4$	$h = 128, l = 1, \text{heads} = 4$	$h = 64, l = 1, \text{heads} = 1$	$h = 16, l = 2, \text{heads} = 4$
IFTTP	$h = 64$	$h = 64$	$h = 32$	$h = 64$	$h = 64$
MHP	$h = 8$	$h = 16$	$h = 4$	$h = 16$	$h = 8$
NHP	$h = 128$	$h = 64$	$h = 128$	$h = 128$	$h = 64$
AttNHP	$h = 64, t = 16, l = 2, \text{heads} = 4$	$h = 16, t = 16, l = 2, \text{heads} = 4$	$h = 16, t = 16, l = 3, \text{heads} = 4$	$h = 32, t = 16, l = 3, \text{heads} = 4$	$h = 32, t = 16, l = 2, \text{heads} = 4$
S2P2	$h = 64, p = 128, l = 2$	$h = 128, p = 128, l = 2$	$h = 128, p = 16, l = 4$	$h = 32, p = 16, l = 4$	$h = 32, p = 32, l = 3$

(b) Model architectures for the additional three benchmark datasets in Table 2

Model	Last.fm	MIMIC-II	EHRSHOT
RMTTP	$h = 256$	$h = 128$	$h = 16$
SAHP	$h = 136, l = 2, \text{heads} = 4$	$h = 64, l = 2, \text{heads} = 4$	$h = 8, l = 2, \text{heads} = 4$
THP	$h = 48, l = 2, \text{heads} = 4$	$h = 32, l = 3, \text{heads} = 4$	$h = 32, l = 2, \text{heads} = 4$
IFTTP	$h = 48$	$h = 256$	$h = 16$
MHP	$h = 16$	$h = 16$	$h = 32$
NHP	$h = 112$	$h = 128$	$h = 80$
AttNHP	$h = 28, t = 16, l = 2, \text{heads} = 4$	$h = 64, t = 16, l = 3, \text{heads} = 2$	OOM
S2P2	$h = 68, p = 16, l = 2$	$h = 64, p = 16, l = 2$	$h = 128, p = 32, l = 2$

C.2 Dataset Statistics

We report the statistics of all eight datasets we used in Table 7. We used the HuggingFace version of the five EasyTPP datasets. For all datasets, we further ensure the MTPP modeling assumptions are satisfied that no more than two events occur at the same time (i.e., inter-arrival time is strictly positive), and event times do not lie on grid points that are effectively discrete-time events. Dataset descriptions and pre-processing details are provided in Appendix C.3.

Table 7: Statistics of the eight datasets we experiment with.

Dataset	K	Number of Events			Sequence Length			Number of Sequences		
		Train	Valid	Test	Min	Max	Mean	Train	Valid	Test
Amazon	16	288,377	40,995	84,048	14	94	44.8	6,454	922	1,851
Retweet	3	2,176,116	215,521	218,465	50	264	108.8	20,000	2,000	2,000
Taxi	10	51,584	7,404	14,820	36	38	37.0	1,400	200	400
Taobao	17	73,483	11,472	28,455	28	64	56.7	1,300	200	500
StackOverflow	22	90,497	25,762	26,518	41	101	64.8	1,401	401	401
Last.fm	120	1,534,738	344,542	336,676	6	501	207.2	7,488	1,604	1,604
MIMIC-II	75	9,619	1,253	1,223	2	33	3.7	2600	325	325
EHRSHOT	668	759,141	165,237	170,147	5	3,955	177.0	4,329	927	927

C.3 Dataset Pre-processing

We use the default train/validation/test splits for EasyTPP benchmark datasets. For MIMIC-II, we copy Du et al. [2016] and keep the 325 test sequences in the test split, and further split the 2,935 training sequences into 2,600 for training and 325 for validation. In our pre-processed datasets, Last.fm and EHRSHOT, we randomly partition into subsets containing 70%, 15%, 15% of all sequences for training/validation/test respectively. We provide a high-level description of all the datasets we used, followed by our pre-processing procedure of Last.fm and EHRSHOT in more detail. Note that for datasets that contain concurrent events or effectively discrete times (e.g., StackOverflow, Retweet), we apply a small amount of jittering to ensure no modeling assumptions are violated in the MTPP framework.

Amazon [Ni et al., 2019] contains user product reviews where product categories are considered as marks. **Retweet** [Zhao et al., 2015] predicts the popularity of a retweet cascade, where the event type is decided by if the retweet comes from users with “small”, “medium”, or “large” influences, measured by number of followers [Mei and Eisner, 2017]. **Taxi** data [Whong, 2014, Mei et al., 2019] uses data from the pickups and dropoffs of New York taxi and the marks are defined as the Cartesian product of five discrete locations and two actions (pickup/dropoff). **Taobao** [Xue et al., 2022] describes the viewing patterns of users on an e-commerce site, where item categories are considered as marks. **StackOverflow** contains the badges (defined as marks) awarded to users on a question-answering website. Finally, **MIMIC-II** [Saeed et al., 2002] records different diseases (used as marks) during hospital visits of patients. We add a small amount of noise to the MIMIC-II event times so that events do not lie on a fixed grid. Both StackOverflow and MIMIC-II datasets were first pre-processed by Du et al. [2016].

Last.fm [Celma Herrada et al., 2009], [McFee et al., 2012] records 992 users’ music listening habits that has been widely used in MTPP literature [Kumar et al., 2019, Boyd et al., 2020, Bosser and Taieb, 2023]. Mark types are defined as the genres of a song, and each event is a play of a particular genre. Each sequence represents the monthly listening behavior of each user, with sequence lengths between 5 and 500. If the song is associated with multiple genres we select a random one of the genres, resulting in a total of 120 different marks.

EHRSHOT [Wornow et al., 2023] is a newly proposed large dataset of longitudinal de-identified patient medical records, and has rich information such as hospital visits, procedures, and measurements. We introduce an MTPP dataset derived from EHRSHOT, where medical services and procedures are treated as marks, as identified by *Current Procedural Terminology* (CPT-4) codes. Each patient defines an event sequence, and we retain only CPT-4 codes with at least 100 occurrences in the dataset. For the $< 1\%$ events of events where there are more than 10 codes at a single timestamp, we retain the top 10 codes with the most frequencies and discard the rest. We then add a small amount of random noise to the event time to ensure they are not overlapping. This process ensures we still satisfy the MTPP framework, and can reasonably instead compute top-10 accuracy for the next mark prediction; other work has considered extending the MTPP framework to consider simultaneous event

occurrence [Chang et al., 2024]. Then we standardize each sequence to start at $t = 0$ and pad the start and end of sequences with a specific padded event token. Note that we do not score these events. Event times are normalized to be in hours. We discard sequences that have less than 5 events or a single timestamp. This leads to the final version of our dataset having 668 marks, and the sequence lengths range from 5 to 3955 events, reflecting patient histories that can span multiple years.

We include code in our forked repository for preparing the EHRSHOT event sequence dataset from the raw EHRSHOT dataset. Note that we cannot distribute the raw data (or derivative dataset) under the terms of the original EHRSHOT dataset requiring credentialed access through PhysioNet.

D Additional Experimental Results

D.1 Full Results on Benchmark Datasets

We provide the full log-likelihood results and corresponding plots in Table 8 and Fig. 6 respectively, where we decompose the likelihood into time and mark likelihoods. The improvement of our S2P2 model is mainly driven by better modeling of time, though we also often obtain best- or second-best predictive performance on marks from the next event prediction accuracy results conditioned on true event time in Table 2c. In aggregate, our model achieves a 1.33 per-event likelihood ratio between itself and the next best method across all datasets (a 33% improvement in likelihood). This is calculated by computing the mean log-likelihood ratio across all datasets and then exponentiating. Doing so is equivalent to taking the geometric mean across likelihood ratios.

Configuration and training details of all models can be found in Appendix C.1. As discussed in Section 7 and grouped in the results, models with continuous-time hidden states can present a richer class of intensities and often empirically outperform those with discrete hidden states. Note for the RMSE results in Table 2b, we follow Mei and Eisner [2017] and use the expected next event time as next event time predictions to minimize the Bayes risk. Unlike them, however, we estimate these with the trapezoidal rule rather than Monte-Carlo simulation via the thinning algorithm. In practice, we have found this to produce an estimator with much lower variance and be faster due to being more readily parallelizable. This stands in contrast to the thinning algorithm which has more hyperparameters (e.g., dominating rate, sampling boundary) that can exacerbate bias.

Table 8: Complete per event log-likelihood results on the held-out test for the eight benchmark datasets we consider, averaged over 5 random seeds. In Table 8a we show the full log-likelihood. We then decompose this log-likelihood into the log-likelihood of the event time in Table 8b, and the time-conditional log-likelihood of the mark type in Table 8c. OOM indicates out of memory; standard deviation in parentheses. We box the best-performing model and underline the second-best. We also report the average rank of models across datasets as a summary metric (lower is better). S2P2 is consistently the best or second best-performing model across all datasets.

(a) Full log-likelihood results (equal to the summation of Table 8b and Table 8c). Extended version of Table 2a

Model	Per Event Log-Likelihood, $\mathcal{L}_{\text{Total}}$ (nats) (\uparrow)								Avg. Ranking (\downarrow)
	Amazon	Retweet	Taxi	Taobao	StackOverflow	Last.fm	MIMIC-II	EHRSHOT	
RMTTP	-2.136 (0.003)	-7.098 (0.217)	0.346 (0.002)	1.003 (0.004)	-2.480 (0.019)	-1.780 (0.005)	-0.472 (0.026)	-8.081 (0.025)	7.1
SAHP	-2.074 (0.029)	-6.708 (0.029)	0.298 (0.057)	1.168 (0.029)	-2.341 (0.058)	-1.646 (0.083)	-0.677 (0.072)	-6.804 (0.126)	5.8
THP	-2.096 (0.002)	-6.659 (0.007)	0.372 (0.002)	0.790 (0.002)	-2.338 (0.014)	-1.712 (0.011)	-0.577 (0.011)	-7.208 (0.096)	6.1
IFTTP	0.496 (0.002)	-10.344 (0.016)	0.453 (0.002)	<u>1.318</u> (0.017)	-2.233 (0.009)	<u>-0.492</u> (0.017)	0.317 (0.052)	-6.596 (0.240)	3.0
MHP	-2.091 (0.002)	-6.564 (0.015)	0.370 (0.008)	0.636 (0.004)	-2.346 (0.012)	-1.676 (0.004)	-0.351 (0.012)	-7.206 (0.407)	5.9
NHP	0.129 (0.012)	<u>-6.348</u> (0.000)	0.514 (0.004)	1.157 (0.004)	-2.241 (0.002)	-0.574 (0.011)	0.060 (0.017)	-3.966 (0.058)	3.0
AttNHP	0.484 (0.077)	-6.499 (0.028)	0.493 (0.009)	1.259 (0.022)	-2.194 (0.016)	-0.592 (0.051)	-0.170 (0.077)	OOM	3.1
S2P2 (Ours)	<u>0.781</u> (0.011)	-6.365 (0.003)	<u>0.522</u> (0.004)	1.304 (0.039)	<u>-2.163</u> (0.009)	-0.557 (0.046)	<u>0.919</u> (0.069)	<u>-2.512</u> (0.369)	<u>1.4</u>

(b) Per event log-likelihood of the event times (higher is better).

Model	Per Event Next Event Time Log-Likelihood, $\mathcal{L}_{\text{Time}}$ (nats) (\uparrow)								Avg. Ranking (\downarrow)
	Amazon	Retweet	Taxi	Taobao	StackOverflow	Last.fm	MIMIC-II	EHRSHOT	
RMTTP	0.011 (0.001)	-6.191 (0.083)	0.622 (0.002)	2.428 (0.004)	-0.797 (0.005)	0.256 (0.007)	-0.188 (0.016)	-1.913 (0.025)	6.1
SAHP	0.115 (0.049)	-5.872 (0.062)	0.645 (0.044)	2.604 (0.008)	-0.703 (0.031)	0.489 (0.078)	-0.244 (0.040)	-1.801 (0.049)	4.9
THP	-0.068 (0.002)	-5.874 (0.007)	0.621 (0.002)	2.242 (0.002)	-0.772 (0.006)	0.220 (0.010)	-0.271 (0.004)	-1.921 (0.027)	7.0
IFTTP	2.483 (0.001)	-9.500 (0.011)	<u>0.735</u> (0.002)	2.708 (0.018)	-0.662 (0.007)	<u>1.277</u> (0.016)	0.555 (0.050)	-2.640 (0.115)	3.1
MHP	-0.064 (0.002)	-5.774 (0.016)	0.620 (0.006)	2.093 (0.004)	-0.761 (0.006)	0.230 (0.003)	-0.140 (0.008)	-2.119 (0.318)	6.4
NHP	2.116 (0.009)	<u>-5.584</u> (0.001)	0.727 (0.003)	2.578 (0.006)	-0.699 (0.002)	1.198 (0.006)	0.225 (0.016)	-0.821 (0.045)	3.3
AttNHP	2.416 (0.092)	-5.726 (0.027)	0.714 (0.010)	2.654 (0.007)	-0.684 (0.005)	1.203 (0.015)	0.031 (0.055)	OOM	3.3
S2P2 (Ours)	<u>2.652</u> (0.009)	-5.598 (0.002)	0.733 (0.003)	<u>2.719</u> (0.038)	<u>-0.641</u> (0.003)	1.257 (0.022)	<u>1.050</u> (0.065)	<u>0.382</u> (0.362)	<u>1.4</u>

(c) Per event log-likelihood of mark type conditioned on the arrival time (higher is better).

Model	Per Event Next Mark Log-Likelihood, $\mathcal{L}_{\text{Mark}}$ (nats) (\uparrow)								Avg. Ranking (\downarrow)
	Amazon	Retweet	Taxi	Taobao	StackOverflow	Last.fm	MIMIC-II	EHRSHOT	
RMTTP	-2.147 (0.003)	-0.908 (0.141)	-0.276 (0.000)	-1.425 (0.002)	-1.683 (0.015)	-2.035 (0.004)	-0.284 (0.014)	-6.168 (0.025)	6.8
SAHP	-2.189 (0.030)	-0.836 (0.036)	-0.346 (0.024)	-1.436 (0.027)	-1.638 (0.032)	-2.136 (0.070)	-0.433 (0.031)	-5.003 (0.132)	6.9
THP	-2.028 (0.002)	-0.785 (0.001)	-0.249 (0.001)	-1.451 (0.000)	-1.566 (0.008)	-1.932 (0.006)	-0.306 (0.009)	-5.287 (0.107)	5.5
IFTTP	-1.988 (0.001)	-0.844 (0.007)	-0.282 (0.001)	<u>1.391</u> (0.005)	-1.571 (0.003)	<u>1.769</u> (0.004)	-0.239 (0.002)	-3.956 (0.192)	4.1
MHP	-2.027 (0.001)	-0.790 (0.003)	-0.251 (0.003)	-1.456 (0.005)	-1.586 (0.006)	-1.906 (0.002)	-0.210 (0.005)	-5.087 (0.296)	5.4
NHP	-1.987 (0.003)	<u>-0.764</u> (0.000)	-0.213 (0.002)	-1.421 (0.004)	-1.542 (0.001)	-1.772 (0.006)	-0.165 (0.002)	-3.144 (0.016)	2.4
AttNHP	-1.933 (0.024)	-0.773 (0.003)	-0.221 (0.002)	-1.395 (0.016)	<u>1.510</u> (0.013)	-1.795 (0.037)	-0.201 (0.025)	OOM	2.4
S2P2 (Ours)	<u>-1.871</u> (0.002)	-0.767 (0.000)	<u>-0.211</u> (0.002)	-1.415 (0.005)	-1.521 (0.008)	-1.814 (0.025)	<u>-0.131</u> (0.014)	<u>-2.893</u> (0.009)	<u>1.9</u>

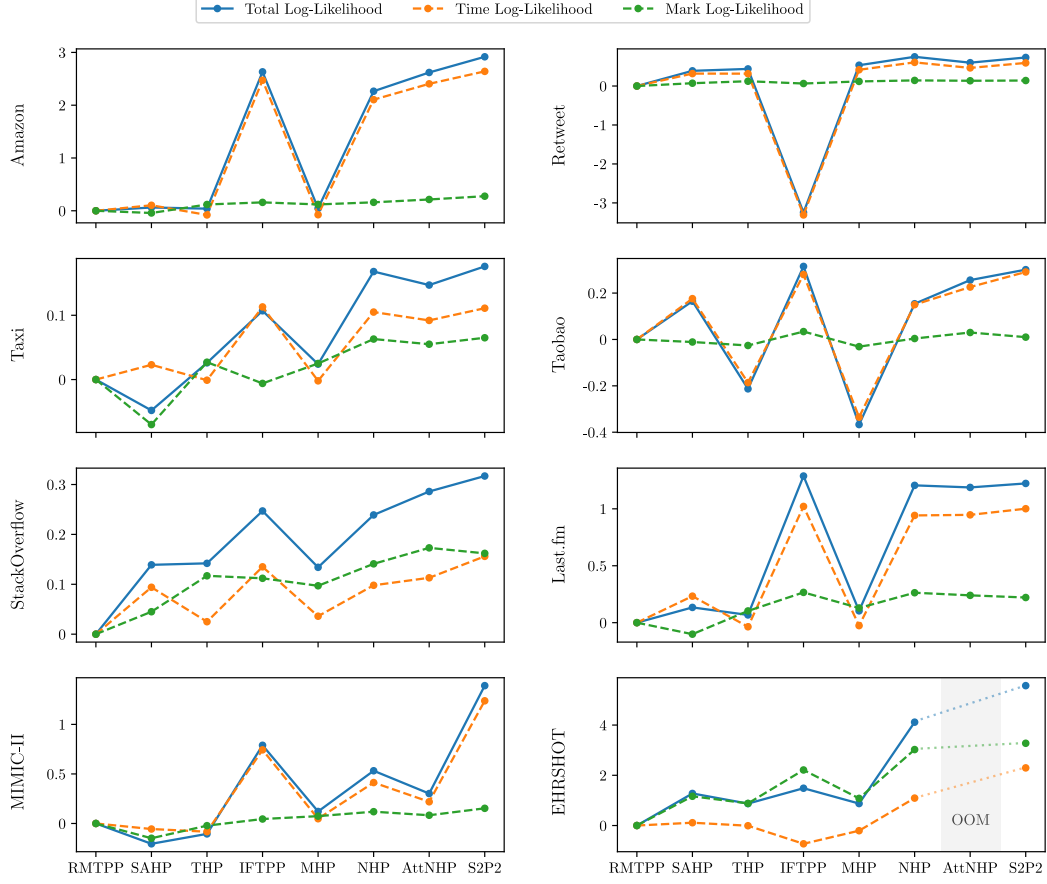


Figure 6: Visualization of $\mathcal{L}_{\text{Total}}$ decomposed into $\mathcal{L}_{\text{Time}}$ and $\mathcal{L}_{\text{Mark}}$ for all models and all datasets relative to RMTPP, normalized by number of events, as discussed in Section 6. The improvement of S2P2 is mainly driven by better modeling of $\mathcal{L}_{\text{Time}}$, while it improves both $\mathcal{L}_{\text{Time}}$ and $\mathcal{L}_{\text{Mark}}$.

D.2 Full Results for Synthetic Poisson Experiments

We present the full results in Fig. 7 for all models regarding the synthetic Poisson experiments discussed in Section 5. All models are trained until convergence using a set of 5,000 generated sequences, where we use 20 Monte Carlo points per event to estimate the integral of log-likelihood during training to accommodate the sparsity of events. We used small models so they do not overfit; model architecture and parameter counts are reported in Table 9. We plot the estimated intensity conditioned on empty sequences using 1,000 equidistant grid points between the start and end points. Our model is the only one that perfectly recovers the underlying ground truth intensity, while also using the fewest parameters.

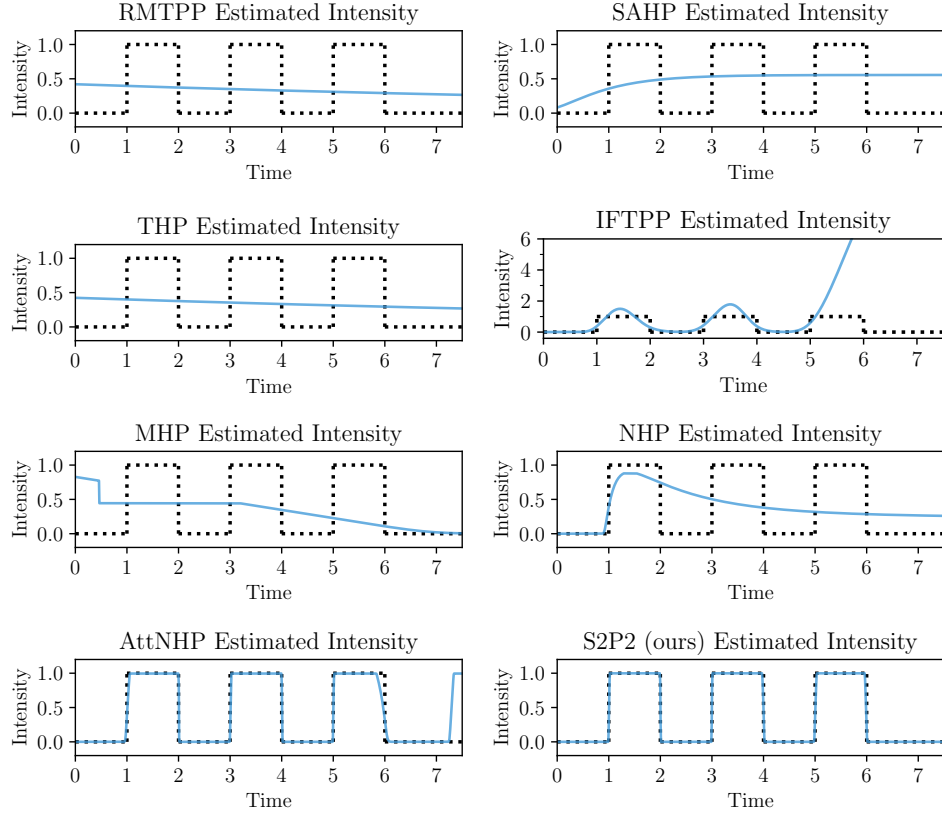


Figure 7: Results for all baseline models for the synthetic Poisson experiment introduced in Section 5. The estimated intensity (blue lines) conditioned on an empty sequence are plotted against the ground truth (dotted black lines).

Table 9: Model architectures and parameter counts for synthetic Poisson experiments.

Model	Architecture	# Parameters
RMTTP	$h = 16$	627
SAHP	$h = 16, l = 2, \text{heads} = 4$	1738
THP	$h = 16, l = 2, \text{heads} = 4$	1684
IFTPP	$h = 16$	1899
MHP	$h = 4$	2240
NHP	$h = 8$	1010
AttNHP	$h = 8, t = 2, l = 2, \text{heads} = 2$	1178
S2P2 (Ours)	$h = 4, p = 4, l = 2$	178

D.3 Additional Synthetic Results on Multivariate Hawkes Processes

We evaluate our model and baseline models against the true model on a randomly initiated parametric Hawkes process with three possible marks. Following the notation in Section 2.1 we draw all parameters from the following distributions: $\nu_i \stackrel{iid}{\sim} \text{Unif}[0.1, 0.5]$, $\alpha_{ij} \stackrel{iid}{\sim} \text{Unif}[0.5, 0.8]$, and $\beta_{ij} \stackrel{iid}{\sim} \text{Unif}[0.4, 1.2]$ for $i, j \in \{1, 2, 3\}$.

All models are trained until convergence using a set of 50,000 generated sequences, where we use 20 Monte Carlo points per event to estimate the integral of log-likelihood during training. Model architecture and parameter counts are reported in Table 10. We plot three example sequences drawn for an additional test set for each model in Fig. 8, using 1,000 equidistant grid points for any inter-event interval. Dotted lines refer to the intensities under the true underlying parametric model; solid lines are different model estimates from trained models.

As we see in inhomogeneous Poisson processes, our model can recover the ground truth intensities with the fewest parameters. Visually, all SAHP, IFTPP, NHP, AttNHP and S2P2 (our model) perform well at recovering the ground truth intensities. It is also worth noting that our model is 7-9 \times quicker than NHP and AttNHP regarding wallclock runtime on a single A5000 GPU. Our results on synthetic experiments validate the model’s ability to recover the ground truth intensities. We further evaluate all models quantitatively using 1,000 test sequences generated from the same multivariate Hawkes process and evaluated both log-likelihood and RMSE for the immediate next event. We see our method competitive again on both metrics.

Table 10: Model architectures and parameter counts for multivariate Hawkes processes experiments.

Model	Architecture	# Parameters
RMTTP	$h = 16$	697
SAHP	$h = 16, l = 2, \text{heads} = 4$	1902
THP	$h = 16, l = 2, \text{heads} = 4$	1756
IFTTP	$h = 16$	1965
MHP	$h = 4$	2264
NHP	$h = 8$	1046
AttNHP	$h = 8, t = 2, l = 2, \text{heads} = 2$	1230
S2P2 (Ours)	$h = 8, p = 4, l = 2$	358

Table 11: Performance comparison of models on the multivariate Hawkes processes experiment presented above. Higher per-event log-likelihood indicates better performance, whereas lower root mean squared error (RMSE) indicates better performance.

Model	Total Log-Likelihood $\mathcal{L}_{\text{Total}} (\uparrow)$	Next-Event Time RMSE (\downarrow)
RMTTP	-0.550	0.648
SAHP	-0.537	0.647
THP	-0.543	0.648
IFTTP	-0.534	0.647
MHP	-0.551	0.648
NHP	-0.530	0.647
AttNHP	-0.533	0.652
S2P2 (Ours)	-0.527	0.647

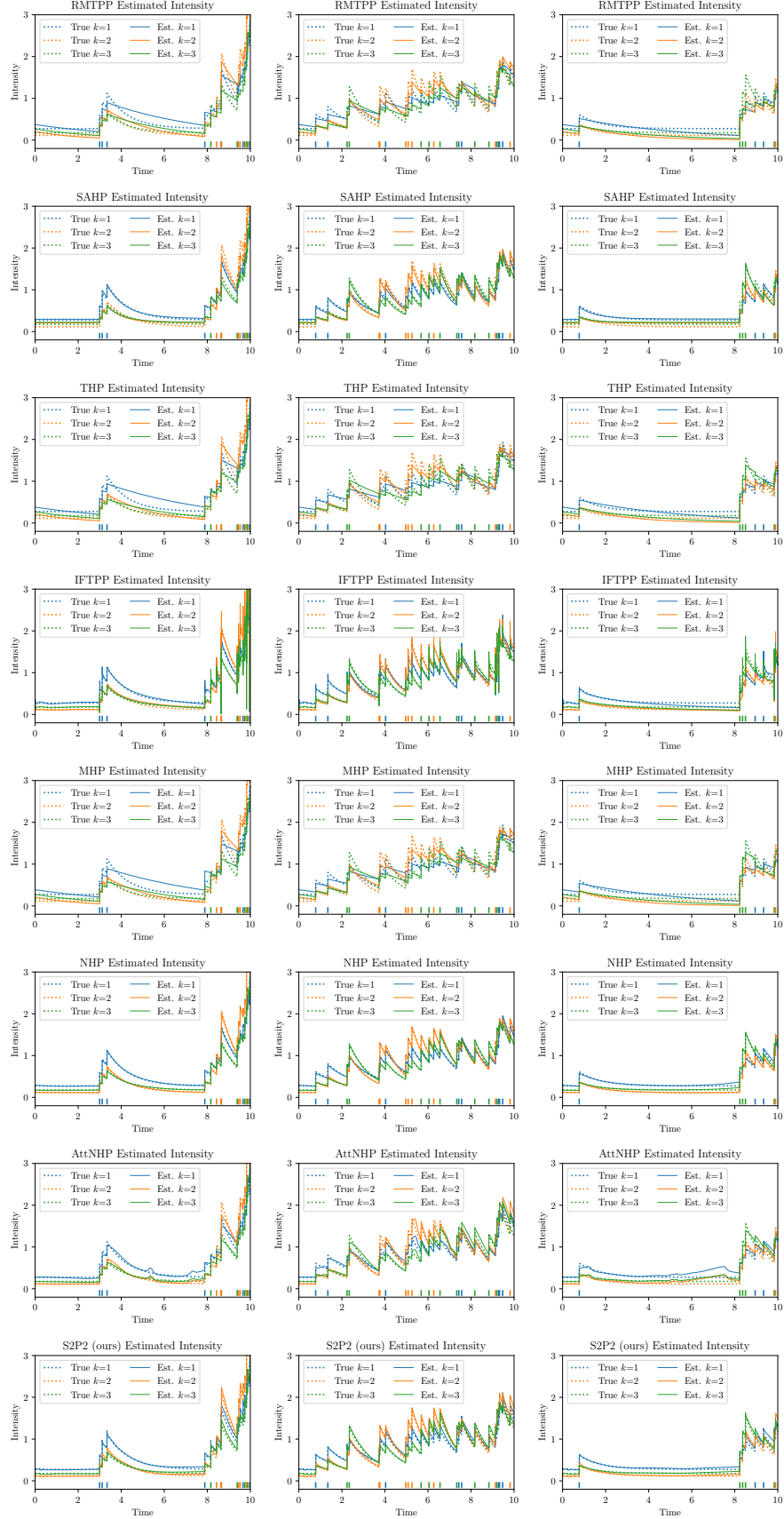



Figure 8: Our proposed S2P2 model and baseline models trained with 50k training sequences drawn from a randomly instantiated multivariate Hawkes process where $K = 3$. For each model, three example test sequences are plotted; the locations of colored bars  indicate the true event times.

D.4 Full Results for Hawkes and Self-Correcting Process

We train our approach on synthetic data generated from known, classical temporal point processes, namely the self-correcting and Hawkes processes. These are characterized by intensity functions $\lambda_t^{\text{SC}} = \exp(t - 0.5N_t)$ and $\lambda_t^{\text{H}} = 0.5 + \sum_{i=1}^{N_t} 0.5 \exp(t_i - t)$, respectively. Models were fit using data drawn from these processes, 6,000 sequences for training and 2,000 for validation. The learned intensity functions, evaluated on a held-out test sequence, can be seen for the self-correcting process data in Fig. 9 and for the Hawkes process data in Fig. 10 for all methods. The hyperparameters for the models were chosen by a grid search (see Table 12). We see many of the models, including ours, do well at capturing the ground truth intensity.

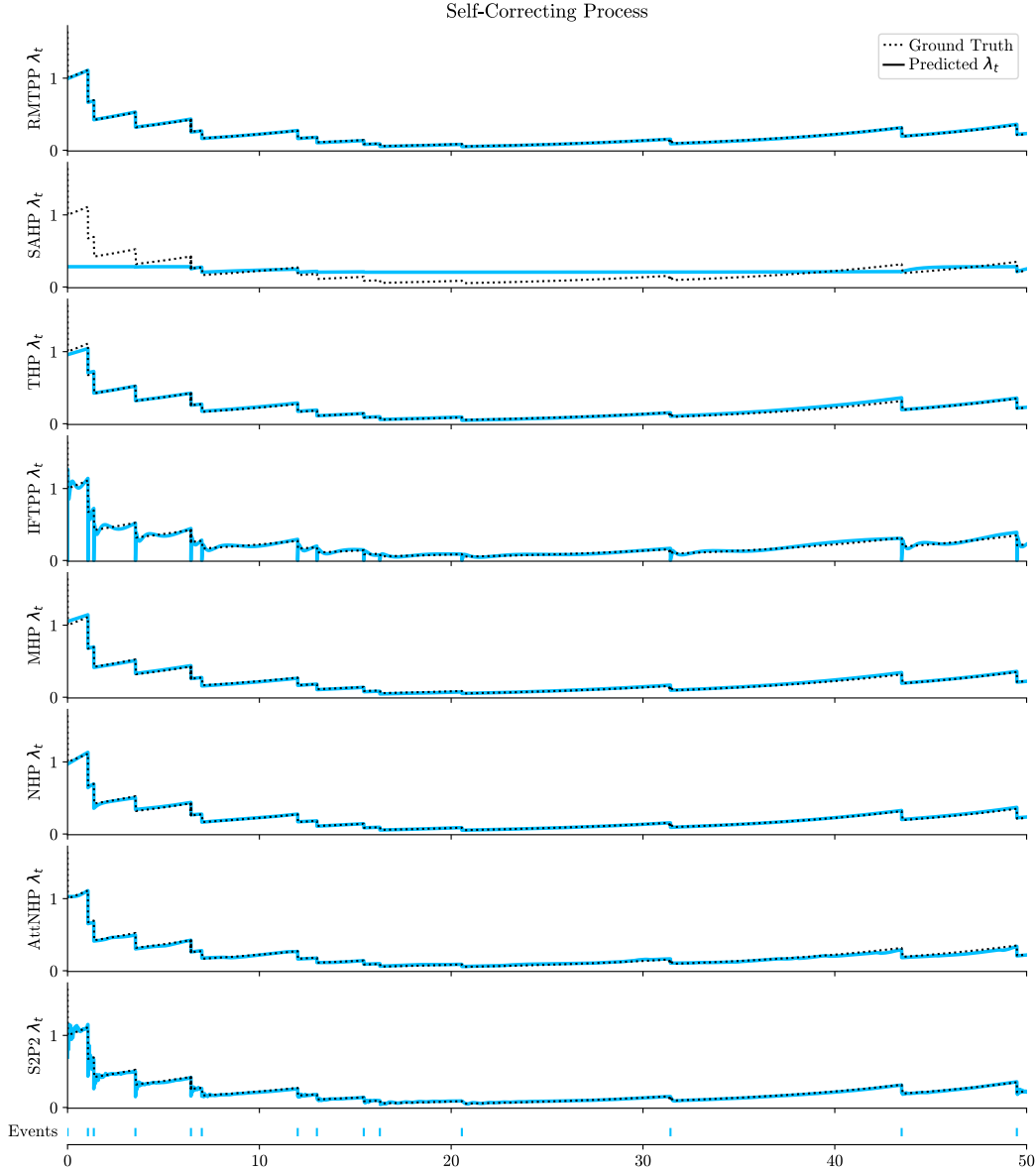


Figure 9: Synthetic self-correcting process experiment visualization of predicted intensities compared to the ground truth intensity for a given held-out sequence. The vertical lines present for IFTPP are due to the conversion from density to intensity being unstable near $\Delta t = 0$.

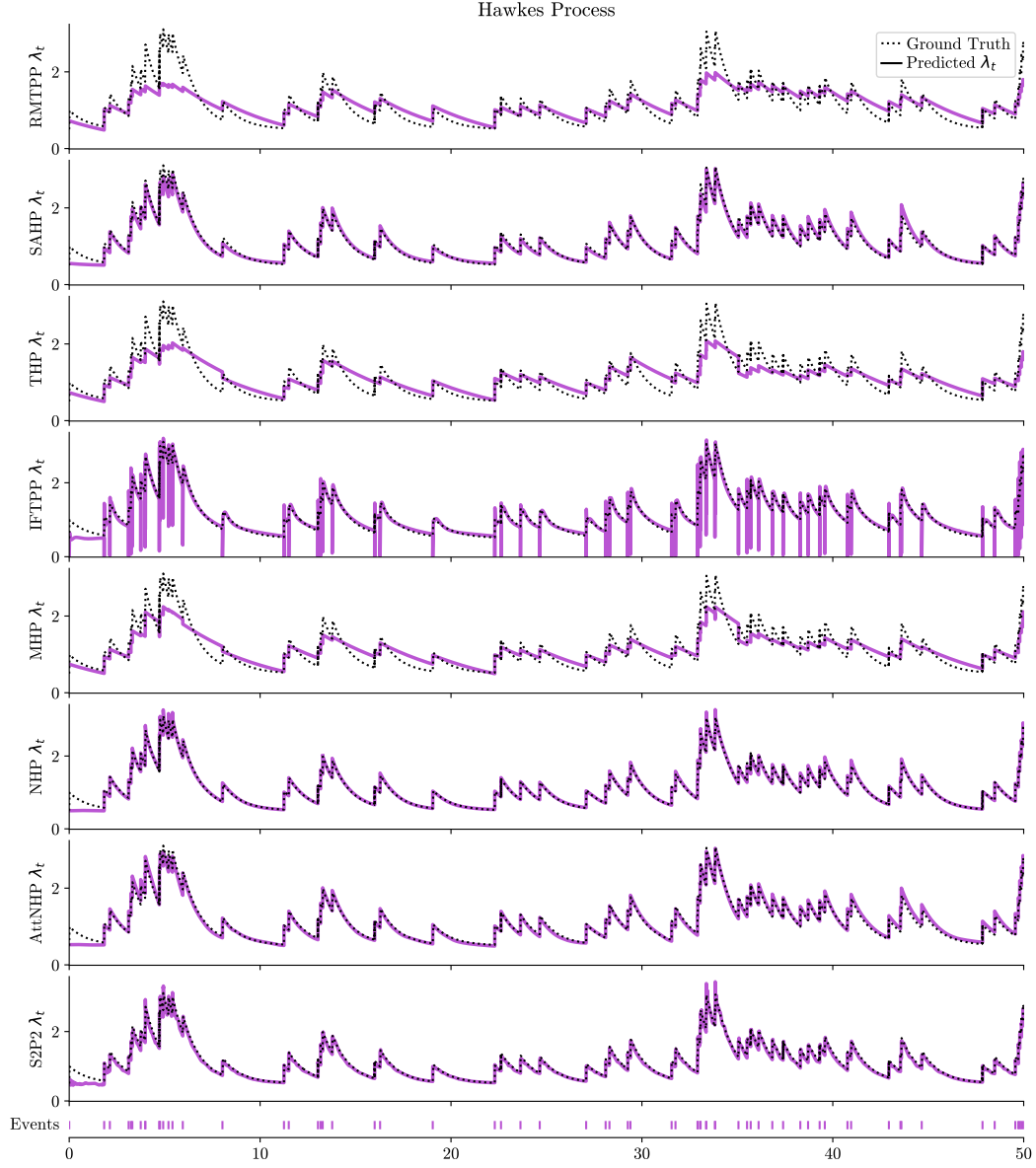


Figure 10: Synthetic Hawkes process experiment visualization of predicted intensities compared to the ground truth intensity for a given held-out sequence. The vertical lines present for IFTPP are due to the conversion from density to intensity being unstable near $\Delta t = 0$.

Table 12: Model architectures and parameter counts for synthetic Hawkes and self-correcting process experiments.

Model	Architecture	# Parameters
RMTTP	$h = 16$	627
SAHP	$h = 16, l = 3, \text{heads} = 4$	2554
THP	$h = 16, l = 3, \text{heads} = 4$	2500
IFTTP	$h = 32$	6859
MHP	$h = 16$	13556
NHP	$h = 32$	14658
AttNHP	$h = 16, t = 16, l = 2, \text{heads} = 4$	13298
S2P2 (Ours)	$h = 16, p = 16, l = 4$	7202

D.5 Full Results for Long-Range Dependency Experiment

To measure the ability to capture long-range dependencies by neural MTPPs, we constructed a generative process with long-range dependencies. For this, we generate sequences over the time window of $[0, 100]$, with three possible marks. The first mark is a “distractor” mark, meaning it has no influence over other events. These events are drawn from a homogeneous Poisson process with rate 1. The second mark is a “trigger” mark, which are directly tied to the third “target” mark. Triggers are also drawn from a homogeneous Poisson process with rate 0.1. For every trigger event $(t_i, k_i = 2)$ drawn, a corresponding target event $(t_j, k_j = 3)$ is generated conditionally independent of all other events according to $t_j | t_i \sim \mathcal{N}(t_i + 40, 0.1)$.

All models were trained with the same hyperparameters as in Table 12. The predicted intensity functions for a single sequence can be seen in Fig. 11 and the likelihood ratio between the trained models and the ground truth process on held-out test sequences can be found in Table 13. We can see that both S2P2 and AttNHP do very well, both qualitatively and quantitatively. This is expected as both architectures are well suited for long-range dependencies while still being continuous-time models, allowing for expressive intensity functions.

Table 13: Likelihood ratios between models and ground truth process for held-out data on long-range experiment.

Model	Ground Truth	RMTTP	SAHP	THP	IFTTP	MHP	NHP	AttNHP	S2P2
Lik. Ratio	100%	80.4%	81.0%	94.0%	88.0%	87.3%	87.9%	<u>99.7%</u>	<u>97.8%</u>

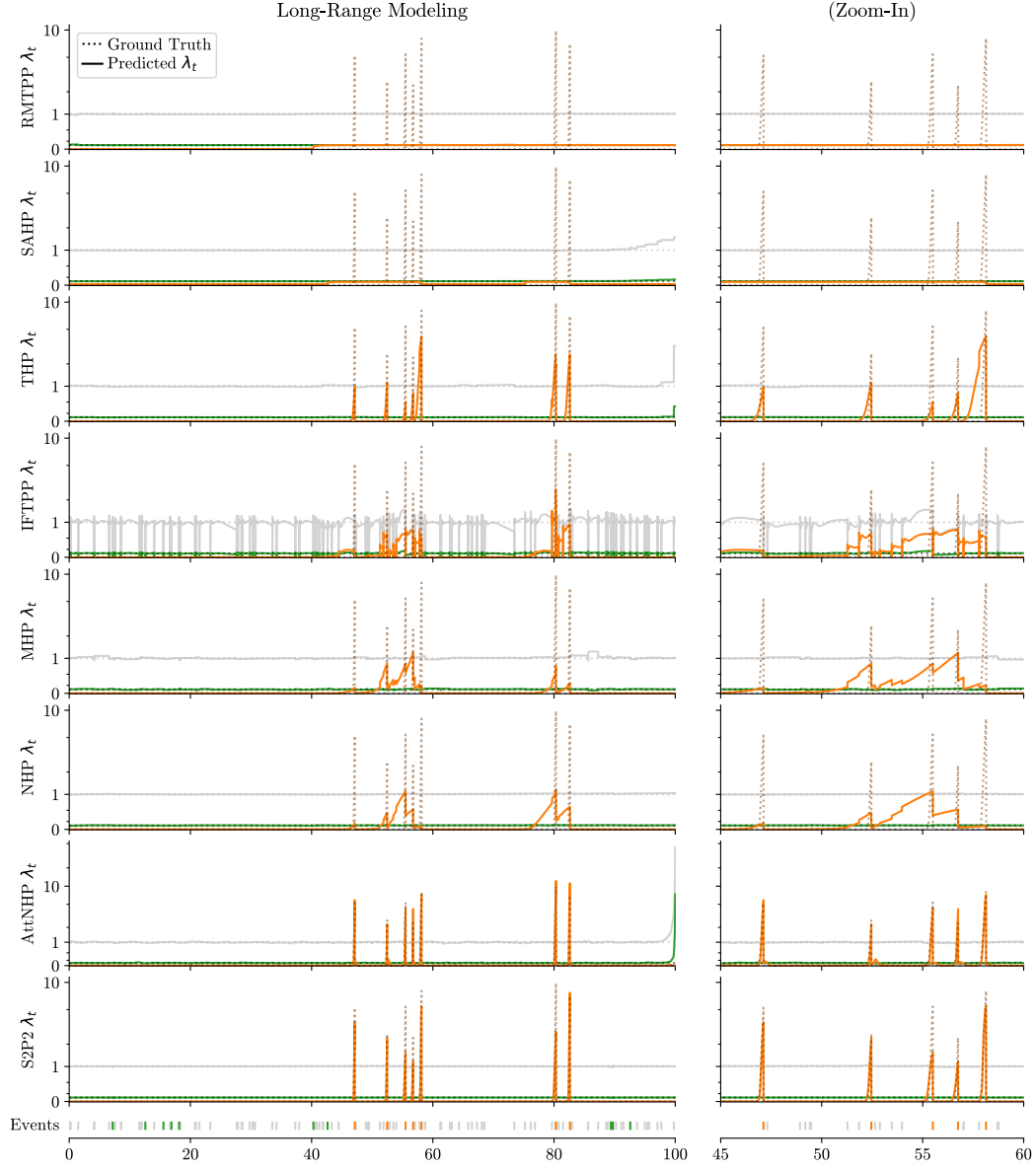


Figure 11: Synthetic long-range experiment visualization of predicted intensities compared to the ground truth intensity for a given held-out sequence. The vertical lines present for IFTPP are due to the conversion from density to intensity being unstable near $\Delta t = 0$.

D.6 Ablation for Different S2P2 Variants

We perform an ablation study of different model variants that we proposed on all datasets and summarize the results in Table 14. We train EHRSHOT using 10% of its training data because larger dataset scale requires more training time (but use the original validation and test sets for model selection and reporting results). Forward and backward discretization are very close in performance, with backwards discretization having a slight edge. Models that are input-dependent achieve better performance on most datasets, although on certain datasets input dependence appears to harm performance. It is an interesting direction for future work to explore theoretically and empirically when each of these variants is best. We select backward discretization with input dependence for the results in the main paper.

Table 14: Ablation for different model variants log-likelihood (LL). ID stands for input-dependent, see Section 3.3. Backward and Forward respectively refer to using \mathbf{u}_{t_i-1} and \mathbf{u}_{t_i-} (i.e., the previous right limit or current left limit), see Appendix B.4.

Dataset	Model variant	LL	Arrival time LL	Mark LL conditioned on time
Amazon	Forward	0.705	2.617	-1.912
	Forward + ID	0.748	2.634	-1.886
	Backward	0.740	2.640	-1.899
	Backward + ID	0.765	2.638	-1.873
Retweet	Forward	-6.405	-5.625	-0.780
	Forward + ID	-6.370	-5.602	-0.767
	Backward	-6.398	-5.618	-0.780
	Backward + ID	-6.367	-5.600	-0.767
Taxi	Forward	0.473	0.697	-0.224
	Forward + ID	0.525	0.733	-0.208
	Backward	0.477	0.705	-0.228
	Backward + ID	0.528	0.738	-0.209
Taobao	Forward	1.207	2.643	-1.435
	Forward + ID	1.332	2.742	-1.410
	Backward	1.215	2.648	-1.432
	Backward + ID	1.332	2.742	-1.410
StackOverflow	Forward	-2.249	-0.676	-1.572
	Forward + ID	-2.174	-0.644	-1.530
	Backward	-2.225	-0.679	-1.547
	Backward + ID	-2.165	-0.636	-1.529
Last.fm	Forward	-0.463	1.309	-1.772
	Forward + ID	-0.477	1.302	-1.779
	Backward	-0.474	1.303	-1.777
	Backward + ID	-0.496	1.294	-1.790
MIMIC-II	Forward	0.555	0.847	-0.292
	Forward + ID	1.319	1.405	-0.086
	Backward	0.322	0.601	-0.279
	Backward + ID	1.231	1.345	-0.114
EHRSHOT (10%)	Forward	-3.885	0.105	-3.990
	Forward + ID	-3.848	-0.021	-3.827
	Backward	-4.571	-0.432	-4.139
	Backward + ID	-4.684	-0.641	-4.043

D.7 Model Calibration

To further probe the models, we evaluate the *calibration* of MTPPs, as proposed by [Bosser and Taieb \[2023\]](#). Calibration has a different focus than log-likelihood-based or accuracy-based evaluation. Calibration instead describes how well the *uncertainty* in the model is reflective of the observed data. It is important to note, however, a model can achieve perfect calibration simply by predicting the marginal distribution. Better calibration therefore *does not* necessarily indicate better predictive performance — only better calibrated errors — and so should be taken in context with the performance under other metrics. We provide summarized statistics for both probabilistic calibration error (PCE) for time calibration and expected calibration error (ECE) for mark calibration in Table [15](#), and visualize the calibration curves in Figs. [12](#) and [13](#).

We see that, on the whole, MTPP models produce fairly well-calibrated predictions. IFTPP is the best calibrated of the models, this may be as a result of having parametric distributions for inter-arrival time (although IFTPP does fail on some datasets such as Retweet). The S2P2 is particularly well calibrated in time (PCE) among intensity-based methods, suggesting again that our S2P2 is capturing time dependencies better than other models. It is also the second-best calibrated on mark prediction (ECE) on average.

Table 15: Calibration results for the models and datasets tests.

(a) Probabilistic calibration error (PCE) for time calibration in percentage.

Model	Probabilistic Calibration Error (PCE) (\downarrow)								Avg. Ranking (\downarrow)
	Amazon	Retweet	Taxi	Taobao	StackOverflow	Last.fm	MIMIC-II	EHRSHOT	
RMTTP	13.67 (0.03)	7.93 (0.62)	3.50 (0.03)	0.22 (0.16)	1.94 (0.10)	1.56 (0.01)	3.63 (0.37)	12.60 (0.37)	6.1
SAHP	12.04 (1.02)	8.51 (1.86)	2.52 (0.99)	3.18 (0.21)	1.50 (0.57)	2.53 (1.86)	2.28 (0.44)	20.20 (1.09)	5.3
THP	12.38 (0.05)	5.68 (0.08)	3.34 (0.02)	6.36 (0.04)	2.06 (0.11)	1.02 (0.08)	<u>1.10</u> (0.06)	13.46 (0.45)	5.4
IFTTP	<u>1.59</u> (0.09)	23.85 (0.26)	<u>0.40</u> (0.10)	<u>1.61</u> (0.74)	<u>0.84</u> (0.34)	<u>0.46</u> (0.44)	1.75 (0.33)	16.58 (3.34)	<u>2.6</u>
MHP	12.22 (0.04)	4.89 (0.16)	3.43 (0.05)	8.77 (0.40)	1.58 (0.13)	1.25 (0.05)	6.21 (0.18)	15.24 (0.92)	6.0
NHP	8.45 (0.28)	<u>0.20</u> (0.19)	0.87 (0.50)	7.40 (0.68)	1.51 (0.11)	4.70 (0.13)	5.92 (0.14)	<u>7.70</u> (0.49)	3.6
AttNHP	6.36 (0.63)	2.09 (0.85)	0.84 (0.27)	3.08 (0.16)	1.65 (0.24)	1.43 (0.14)	4.70 (0.33)	OOM	3.7
S2P2 (Ours)	<u>5.88</u> (0.17)	<u>0.44</u> (0.27)	<u>0.55</u> (0.33)	<u>2.07</u> (0.32)	<u>1.03</u> (0.15)	<u>1.38</u> (0.52)	11.70 (0.68)	<u>12.06</u> (0.54)	<u>2.8</u>

(b) Expected calibration error (ECE) for mark calibration in percentage.

Model	Expected Calibration Error (ECE) (\downarrow)								Avg. Ranking (\downarrow)
	Amazon	Retweet	Taxi	Taobao	StackOverflow	Last.fm	MIMIC-II	EHRSHOT	
RMTTP	6.58 (0.15)	3.99 (4.28)	2.42 (0.16)	1.89 (0.24)	2.10 (0.27)	2.47 (0.45)	2.79 (0.43)	8.47 (0.31)	5.8
SAHP	8.17 (2.00)	6.27 (2.23)	6.77 (0.21)	2.68 (0.35)	1.71 (0.77)	6.26 (4.30)	5.41 (0.26)	5.85 (1.95)	6.8
THP	2.06 (0.17)	1.26 (0.11)	1.76 (0.07)	6.51 (0.03)	<u>0.81</u> (0.14)	3.42 (0.70)	2.16 (0.39)	8.95 (0.91)	5.0
IFTTP	<u>0.46</u> (0.10)	0.95 (1.12)	<u>0.55</u> (0.19)	<u>1.20</u> (0.20)	1.28 (0.54)	0.66 (0.05)	<u>1.39</u> (0.23)	<u>1.99</u> (0.61)	<u>1.8</u>
MHP	1.65 (0.16)	1.18 (0.12)	1.91 (0.11)	4.15 (0.36)	0.82 (0.18)	2.83 (0.50)	2.22 (0.24)	10.00 (1.71)	4.8
NHP	8.30 (0.21)	<u>0.35</u> (0.06)	0.79 (0.10)	5.59 (0.69)	1.31 (0.16)	3.41 (0.41)	2.24 (0.32)	4.18 (0.69)	4.9
AttNHP	3.13 (0.61)	0.52 (0.16)	0.56 (0.10)	2.47 (0.12)	1.37 (0.42)	<u>0.61</u> (0.16)	2.23 (0.50)	OOM	3.4
S2P2 (Ours)	<u>0.88</u> (0.34)	<u>0.52</u> (0.13)	0.58 (0.12)	1.96 (0.67)	1.98 (0.19)	1.01 (0.63)	<u>1.62</u> (0.24)	<u>2.51</u> (0.44)	<u>3.0</u>

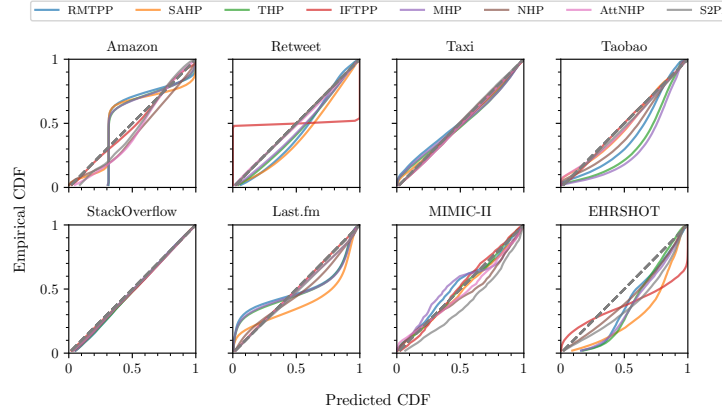


Figure 12: Reliability diagram for predicted inter-arrival time for each model on all datasets. Diagonal dashed lines refer to perfect calibration.

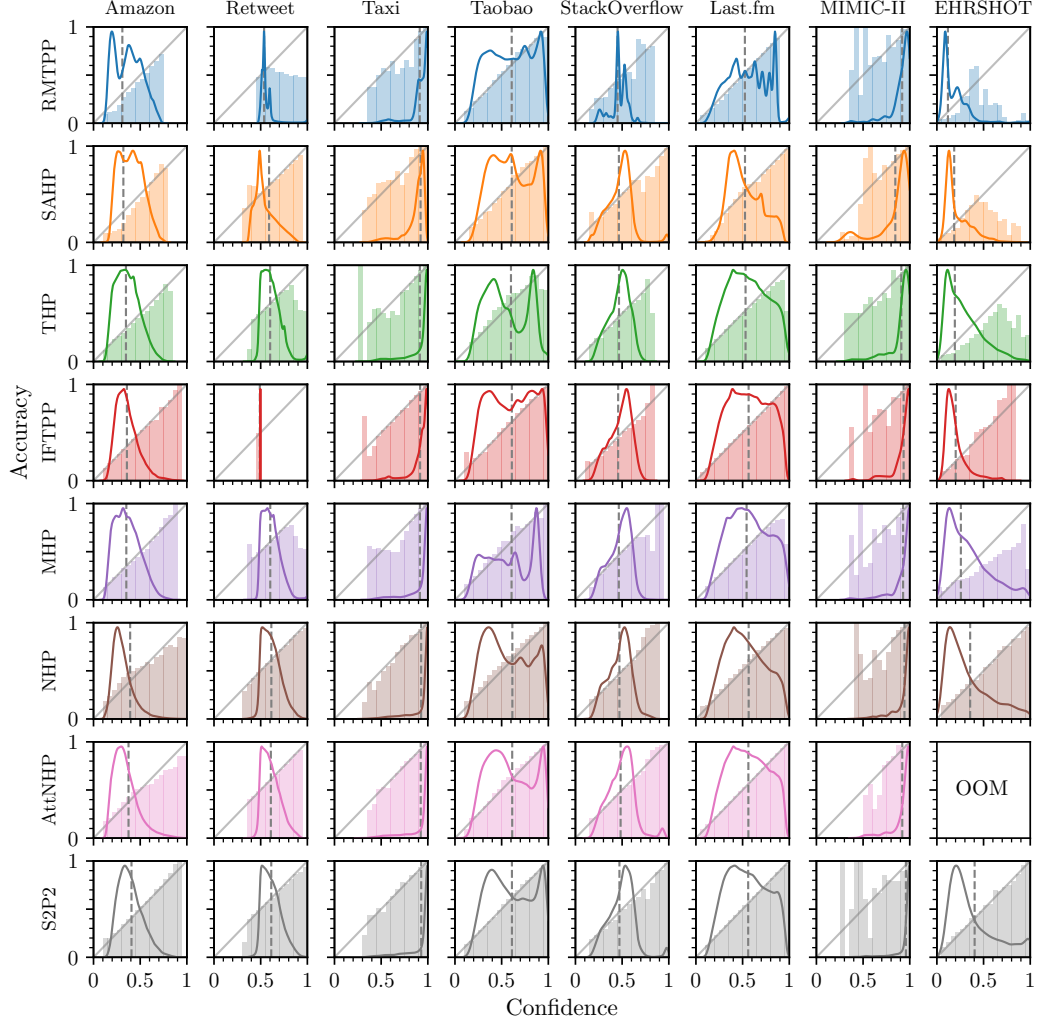


Figure 13: Reliability diagram for mark prediction of all models and all datasets. The x -axis specifies the confidence of model estimates grouped into 20 bins, and the y -axis of the bar plot is the model accuracy within that bin. The diagonal lines represent perfect calibration. The solid curves depict the distribution of confidences, and do not share the y -axis. The grey dashed lines indicate the overall prediction accuracy of the model for the next event conditioned on true event time.

Finally, in Figs. 14 and 15 we plot the log-likelihood of time and mark respectively, versus their corresponding calibration results, to provide an overall view of the performances of different models. Our S2P2 model consistently achieves higher log-likelihood while maintaining good calibration on both time and mark components on most datasets.

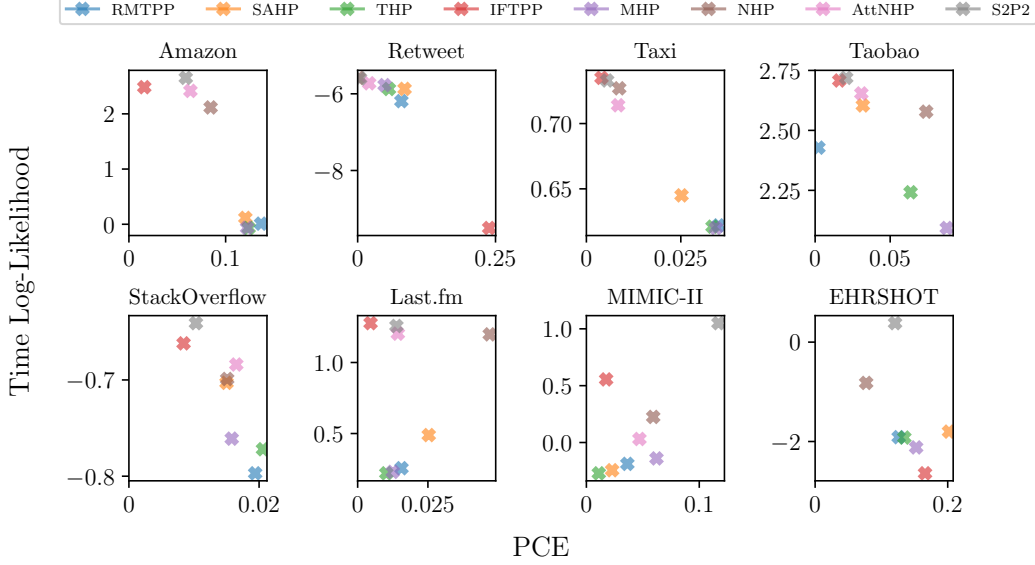


Figure 14: Log-likelihood of time vs. PCE for all models grouped by datasets. Higher log-likelihood and lower PCE are better (i.e., top left corner).

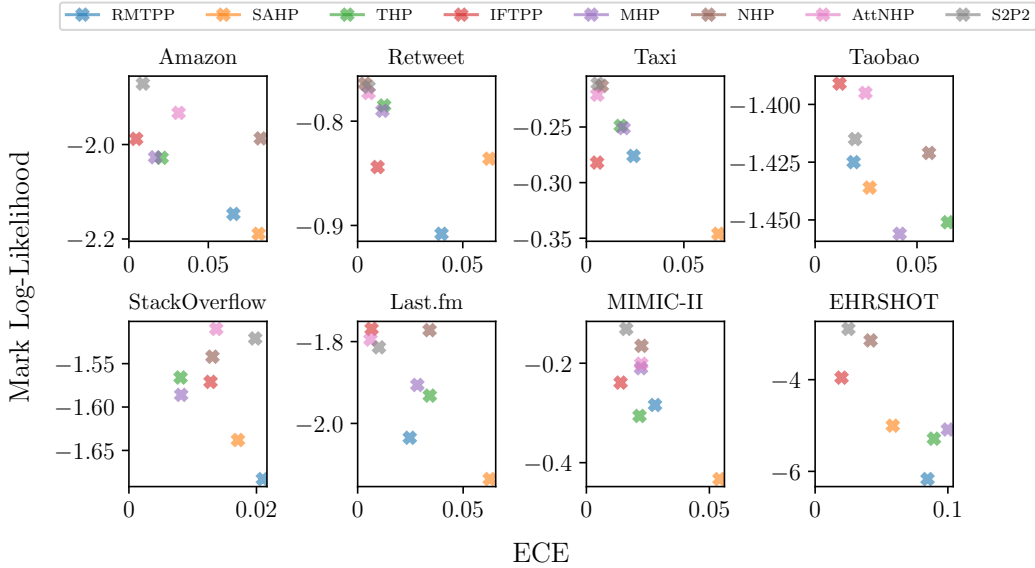


Figure 15: Log-likelihood of mark vs. ECE for all models grouped by datasets. Higher log-likelihood and lower ECE are better (i.e., top left corner).

References

- Ali Caner Türkmen, Yuyang Wang, and Tim Januschowski. Intermittent demand forecasting with deep renewal processes. *arXiv preprint arXiv:1911.10416*, 2019.
- Bjørnar Vassøy, Massimiliano Ruocco, Eliezer de Souza da Silva, and Erlend Aune. Time is of the essence: a joint hierarchical RNN and point process model for time and item predictions. In *Proceedings of the twelfth ACM international conference on Web search and data mining*, pages 591–599, 2019.
- Guolei Yang, Ying Cai, and Chandan K Reddy. Recurrent spatio-temporal point process for check-in time prediction. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 2203–2211, 2018.
- William Hua, Hongyuan Mei, Sarah Zohar, Magali Giral, and Yanxun Xu. Personalized dynamic treatment regimes in continuous time: a Bayesian approach for optimizing clinical decisions with timing. *Bayesian Analysis*, 17(3):849–878, 2022.
- Álvaro Gajardo and Hans-Georg Müller. Point process models for COVID-19 cases and deaths. *Journal of Applied Statistics*, 50(11-12):2294–2309, 2023.
- Alex Williams, Anthony Degleris, Yixin Wang, and Scott Linderman. Point process models for sequence detection in high-dimensional neural spike trains. *Advances in neural information processing systems*, 33:14350–14361, 2020.
- Anuj Sharma, Robert Johnson, Florian Engert, and Scott Linderman. Point process latent variable models of larval zebrafish behavior. *Advances in Neural Information Processing Systems*, 31, 2018.
- Jianlong Wang, Xiaoqi Duan, Peixiao Wang, A-Gen Qiu, and Zeqiang Chen. Predicting urban signal-controlled intersection congestion events using spatio-temporal neural point process. *International Journal of Digital Earth*, 17(1):2376270, 2024.
- Nan Du, Hanjun Dai, Rakshit Trivedi, Utkarsh Upadhyay, Manuel Gomez-Rodriguez, and Le Song. Recurrent marked temporal point processes: Embedding event history to vector. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1555–1564, 2016.
- Hongyuan Mei and Jason M Eisner. The neural Hawkes process: A neurally self-modulating multivariate point process. *Advances in Neural Information Processing Systems*, 30:6757–6767, 2017.
- Qiang Zhang, Aldo Lipani, Omer Kirnap, and Emine Yilmaz. Self-attentive Hawkes process. In *International conference on machine learning*, pages 11183–11193. PMLR, 2020.
- Simiao Zuo, Haoming Jiang, Zichong Li, Tuo Zhao, and Hongyuan Zha. Transformer Hawkes process. In *International conference on machine learning*, pages 11692–11702. PMLR, 2020.
- Chenghao Yang, Hongyuan Mei, and Jason Eisner. Transformer embeddings of irregularly spaced events and their participants. In *Proceedings of the tenth international conference on learning representations (ICLR)*, 2022.
- Oleksandr Shchur, Marin Bilos, and Stephan Günnemann. Intensity-free learning of temporal point processes. In *International Conference on Learning Representations*, 2020a.
- Anningzhe Gao, Shan Dai, and Yan Hu. Mamba Hawkes process. *arXiv preprint arXiv:2407.05302*, 2024.
- Albert Gu, Karan Goel, and Christopher Re. Efficiently modeling long sequences with structured state spaces. In *International Conference on Learning Representations*, 2022a.
- Jimmy TH Smith, Andrew Warrington, and Scott Linderman. Simplified state space layers for sequence modeling. In *The Eleventh International Conference on Learning Representations*, 2022.
- Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

- Albert Gu, Isys Johnson, Karan Goel, Khaled Saab, Tri Dao, Atri Rudra, and Christopher Ré. Combining recurrent, convolutional, and continuous-time models with linear state space layers. *Advances in neural information processing systems*, 34:572–585, 2021.
- Karan Goel, Albert Gu, Chris Donahue, and Christopher Ré. It’s raw! audio generation with state-space models. In *International Conference on Machine Learning*, pages 7616–7633. PMLR, 2022.
- Fei Deng, Junyeong Park, and Sungjin Ahn. Facing off world model backbones: RNNs, Transformers, and S4. *Advances in Neural Information Processing Systems*, 36, 2024.
- Alan G Hawkes. Spectra of some self-exciting and mutually exciting point processes. *Biometrika*, 58 (1):83–90, 1971.
- Daryl J Daley and David Vere-Jones. *An Introduction to the Theory of Point Processes: Volume I: Elementary Theory and Methods*. Springer, 2003.
- Guy Blelloch. Prefix sums and their applications. Technical report, Tech. rept. CMU-CS-90-190. School of Computer Science, Carnegie Mellon, 1990.
- Arieh Iserles. *A first course in the numerical analysis of differential equations*. 44. Cambridge university press, 2009.
- Albert Gu, Tri Dao, Stefano Ermon, Atri Rudra, and Christopher Ré. HiPPO: Recurrent memory with optimal polynomial projections. *Advances in neural information processing systems*, 33: 1474–1487, 2020.
- Mark Davis. *Stochastic modelling and control*. Springer Science & Business Media, 2013.
- Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (GELUs). *arXiv preprint arXiv:1606.08415*, 2016.
- Jimmy Lei Ba. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.
- Thomas Josef Liniger. *Multivariate Hawkes processes*. PhD thesis, ETH Zurich, 2009.
- Guilherme Augusto Zagatti, See Kiong Ng, and Stéphane Bressan. Learning multivariate temporal point processes via the time-change theorem. In *International Conference on Artificial Intelligence and Statistics*, pages 3241–3249. PMLR, 2024.
- Felix Draxler, Yang Meng, Kai Nelson, Lukas Laskowski, Yibo Yang, Theofanis Karaletsos, and Stephan Mandt. Transformers for mixed-type event sequences. In *Proceedings of the 38th Conference on Neural Information Processing Systems (NeurIPS)*, 2025.
- Wonho Bae, Mohamed Osama Ahmed, Frederick Tung, and Gabriel L Oliveira. Meta temporal point processes. *arXiv preprint arXiv:2301.12023*, 2023.
- Mai Zeng, Florence Regol, and Mark Coates. Interacting diffusion processes for event sequence forecasting. *arXiv preprint arXiv:2310.17800*, 2023.
- Shuai Zhang, Chuan Zhou, Yang Aron Liu, Peng Zhang, Xixun Lin, and Zhi-Ming Ma. Neural jump-diffusion temporal point processes. In *Forty-first International Conference on Machine Learning*, 2024.
- Ali Caner Türkmen, Yuyang Wang, and Alexander J Smola. Fastpoint: Scalable deep point processes. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2019, Würzburg, Germany, September 16–20, 2019, Proceedings, Part II*, pages 465–480. Springer, 2020.
- Oleksandr Shchur, Nicholas Gao, Marin Biloš, and Stephan Günnemann. Fast and flexible temporal point processes with triangular maps. *Advances in neural information processing systems*, 33: 73–84, 2020b.
- Ankit Gupta, Albert Gu, and Jonathan Berant. Diagonal state spaces are as effective as structured state spaces. *Advances in Neural Information Processing Systems*, 35:22982–22994, 2022.

- Albert Gu, Karan Goel, Ankit Gupta, and Christopher Ré. On the parameterization and initialization of diagonal state space models. *Advances in Neural Information Processing Systems*, 35:35971–35983, 2022b.
- Jue Wang, Wentao Zhu, Pichao Wang, Xiang Yu, Linda Liu, Mohamed Omar, and Raffay Hamid. Selective structured state-spaces for long-form video understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6387–6397, 2023.
- Lianghui Zhu, Bencheng Liao, Qian Zhang, Xinlong Wang, Wenyu Liu, and Xinggang Wang. Vision Mamba: Efficient visual representation learning with bidirectional state space model. *arXiv preprint arXiv:2401.09417*, 2024.
- Nicola Muca Cirone, Antonio Orvieto, Benjamin Walker, Cristopher Salvi, and Terry Lyons. Theoretical foundations of deep selective state-space models. *Advances in Neural Information Processing Systems*, 37:127226–127272, 2024.
- Siqiao Xue, Xiaoming Shi, Zhixuan Chu, Yan Wang, Hongyan Hao, Fan Zhou, Caigao Jiang, Chen Pan, James Y Zhang, Qingsong Wen, et al. EasyTPP: Towards open benchmarking temporal point processes. In *The Twelfth International Conference on Learning Representations*, 2023.
- Michael Wornow, Rahul Thapa, Ethan Steinberg, Jason Fries, and Nigam Shah. EHRSHOT: An ehr benchmark for few-shot evaluation of foundation models. *Advances in Neural Information Processing Systems*, 36:67125–67137, 2023.
- Tanguy Bossier and Souhaib Ben Taieb. On the predictive accuracy of neural temporal point process models for continuous-time event data. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. Survey Certification.
- Ameen Ali Ali, Itamar Zimerman, and Lior Wolf. The hidden attention of Mamba models. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar, editors, *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1516–1534, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.76. URL <https://aclanthology.org/2025.acl-long.76/>.
- William Merrill, Jackson Petty, and Ashish Sabharwal. The illusion of state in state-space models. In *International Conference on Machine Learning*, pages 35492–35506. PMLR, 2024.
- Alex Boyd, Yuxin Chang, Stephan Mandt, and Padhraic Smyth. Inference for mark-censored temporal point processes. In *Uncertainty in Artificial Intelligence*, pages 226–236. PMLR, 2023.
- Pritish Chakraborty, Vinayak Gupta, Srikanta J Bedathur, Abir De, et al. Differentiable adversarial attacks for marked temporal point processes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 15704–15712, 2025.
- Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations (ICLR)*, 2015.
- Jianmo Ni, Jiacheng Li, and Julian McAuley. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pages 188–197, 2019.
- Qingyuan Zhao, Murat A Erdogdu, Hera Y He, Anand Rajaraman, and Jure Leskovec. 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*, pages 1513–1522, 2015.
- Chris Whong. FOILing NYC’s taxi trip data. https://chriswhong.com/open-data/foil_nyc_taxi/, 2014. [Online; accessed Oct 15, 2024].
- Hongyuan Mei, Guanghui Qin, and Jason Eisner. Imputing missing events in continuous-time event streams. In *International Conference on Machine Learning*, pages 4475–4485. PMLR, 2019.

- Siqiao Xue, Xiaoming Shi, James Zhang, and Hongyuan Mei. HYPRO: A hybridly normalized probabilistic model for long-horizon prediction of event sequences. *Advances in Neural Information Processing Systems*, 35:34641–34650, 2022.
- Mohammed Saeed, Christine Lieu, Greg Raber, and Roger G Mark. MIMIC II: a massive temporal icu patient database to support research in intelligent patient monitoring. In *Computers in cardiology*, pages 641–644. IEEE, 2002.
- Òscar Celma Herrada et al. *Music recommendation and discovery in the long tail*. Universitat Pompeu Fabra, 2009.
- Brian McFee, Thierry Bertin-Mahieux, Daniel PW Ellis, and Gert RG Lanckriet. The million song dataset challenge. In *Proceedings of the 21st International Conference on World Wide Web*, pages 909–916, 2012.
- 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 & data mining*, pages 1269–1278, 2019.
- Alex Boyd, Robert Bamler, Stephan Mandt, and Padhraic Smyth. User-dependent neural sequence models for continuous-time event data. *Advances in Neural Information Processing Systems*, 33: 21488–21499, 2020.
- Yuxin Chang, Alex Boyd, and Padhraic Smyth. Probabilistic modeling for sequences of sets in continuous-time. In *International Conference on Artificial Intelligence and Statistics*, pages 4357–4365. PMLR, 2024.