Automatic Integration for Fast and Interpretable Neural Point Processes

Zihao Zhou ZIZ244@UCSD.EDU
University of California, San Diego

Rose Yu ROSEYU@UCSD.EDU
University of California, San Diego

Editors: N. Matni, M. Morari, G. J. Pappas

Abstract

The fundamental bottleneck of learning continuous-time point processes is integration. Due to the intrinsic mathematical difficulty of symbolic integration, neural point process (NPP) models either constrain the intensity function to a simple integrable kernel function or apply numerical integration. However, the former has limited expressive power. The latter suffers additional numerical errors and high computational costs. In this paper, we introduce *Automatic Integration for Neural Point Process* models (`Auto-NPP`), a new paradigm for exact, efficient, non-parametric inference of point process. We validate our method on simulated events governed by temporal point processes and real-world events. We demonstrate that our method has clear advantages in recovering complex intensity functions from irregular time series. On real-world datasets with noise and unknown intensity functions, our method is also much faster than state-of-the-art NPP models with comparable prediction accuracy. Our code and data can be found at https://github.com/Rose-STL-Lab/AutoNPP.

Keywords: temporal dynamics, neural point processes, integration method

1. Introduction

Neural point process (NPP) is a family of deep generative models that integrate deep neural networks (DNN) with point processes for modeling irregularly sampled event data with continuous-time dynamics. NPPs allow hidden states to vary between events and are particularly well-suited for learning the dynamics behind discrete events such as social media posts, disease transmission, and earthquakes. The surging interests in NPP have led to the development of many state-of-the-art time series models such as Neural Hawkes process (Mei and Eisner, 2016), Neural ODE (Chen et al., 2018), as outlined in a recent survey on NPP (Shchur et al., 2021).

A central concept in point processes is the *intensity function*, which captures the expected rates of events occurrence. Specifically, given an event sequence over time $\mathcal{H}_t = \{t_1, t_2, ..., t_n\}|_{t_n<t}$, the joint log-likelihood of the observed events can be defined as follows:

$$\log p(\mathcal{H}_t) = \sum_{i=1}^{n} \log \lambda^*(t_i) - \int_0^t \lambda^*(\tau) d\tau \qquad (1)$$

where $\lambda^\star$ refers to the optimal intensity function that best models the sequence.

A major difficulty of maximum likelihood estimation for point processes lies in the integral term of Equation 1. Obtaining an analytical solution to the likelihood is especially difficult for NPPs with hidden states represented by neural networks. For example, the Neural Hawkes process (Mei and Eisner, 2016) does not have a closed-form solution for the likelihood and must rely on Monte Carlo sampling. Furthermore, existing NPPs often evaluate the test log-likelihood only without verifying the learned intensity function. However, as we will demonstrate later, a relatively high test log-likelihood can provide little information about the temporal patterns of influence. Models with flat or complex intensity functions can have very similar test likelihood, rendering it an inadequate evaluation metric.

On the other hand, NPPs with analytical integral rely on strong assumptions about the intensity and are less expressive. Existing works assume that the current influence follows an exponential decay of the intensity (Du et al., 2016), or of the latent representation (Mozer et al., 2017), or a linear interpolation (Zuo et al., 2020). However, these assumptions may be challenging to meet in the real world. For example, in social media posts, a “delayed jump” in influence can occur where a viral post’s impact will not skyrocket until it is forwarded by an opinion leader. This type of event can produce a jump in intensity that violates the smoothness assumption. Additionally, events can exhibit “cyclic influence”, such as tweets being more influential in the evening than in the afternoon.

To reduce computational cost and improve the expressivity of NPPs, we ask a natural question:

Can we directly use a deep neural network to approximate the influence function?

If successful, the resulting NPP could significantly relax the assumptions imposed by existing NPPs and open up new venues for modeling complex real-world event dynamics, including those with “delayed jump” or “cyclic influence”. However, this would require integrating a complicated neural function over a significant time period, where numerical integration is both inefficient and erroneous.

We propose to leverage automatic integration (Lindell et al., 2021; Li et al., 2019), or *AutoInt*, for NPPs. We recognize that taking the partial derivative of a DNN results in a new computational graph that shares the same parameters, as illustrated in Figure 1. We first construct a monotonically increasing integral network whose partial derivative is the intensity function. Then, we train the integral network to maximize the data likelihood. Finally, we reassemble the parameters of the integral network to obtain the intensity function. With AutoInt, we can fit the exact intensity function and its antiderivative without imposing any constraints on their functional forms.

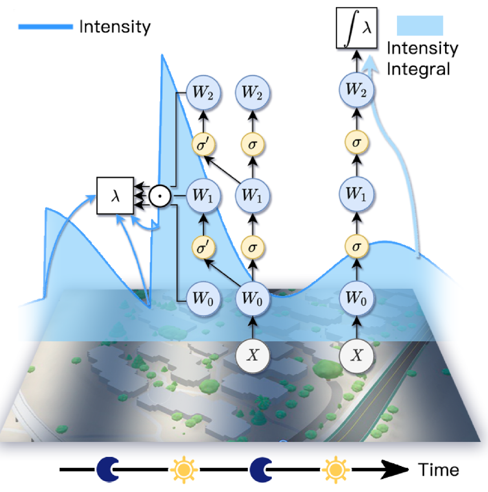

Figure 1: Illustration of automatic integration for NPP. W denotes the linear layer’s weight. σ is the nonlinear activation function. The intensity (λ) network is on the left and the integral ($\int \lambda$) network is on the right. The two networks share the same parameters.

Our approach can efficiently compute the exact likelihood of *any* intensity function. We validate our approach using synthetic temporal point processes with multimodal and discontinuous intensity functions and two real-world datasets. Notably, an earlier work by (Omi et al., 2019) proposes an RNN-based model for the temporal point process, calculating the integral of the intensity function using a similar method. However, our proposed method models the intensity function without recurrent representations, leading to better interpretability and improved ability in recovering complex intensity functions.

To summarize, our contributions are the following:

- We propose an efficient and interpretable framework for neural point processes based on automatic integration. Specifically, we directly approximate the influence function with a neural network and enforce the positivity of the intensity via a monotone integral network.
- We show with synthetic data that our automatic integration framework can faithfully recover complex intensity functions with higher efficiency and accuracy than existing NPP approaches and traditional numerical integration methods.
- On real-world discrete event data, including earthquake Japan, our framework enjoys better interpretability and results in performance comparable or superior to state-of-the-art methods.

2. Related Work

Parametrizing Point Process. Fitting traditional TPP models such as Hawkes process to data points may have lousy performance if the model is misspecified. To address this issue, statisticians have extensively studied non-parametric inference for TPP. Early works usually rely on Bayesian methods (Møller et al., 1998; Kottas and Sansó, 2007; Cunningham et al., 2008). Rathbun and Cressie (1994) modeled the intensity function as a piecewise-constant log Gaussian. Adams et al. (2009) proposed a Markov Chain Monte Carlo (MCMC) inference scheme for the Poisson process with Gaussian priors. These Bayesian models are scalable but assume a continuous intensity change. They can hardly handle the event sequences where the underlying dynamics rapidly change upon event arrivals, such as social media posts.

Recently, NPPs that combine TPP with neural networks has received considerable attention (Yan et al., 2018; Upadhyay et al., 2018; Huang et al., 2019; Shang and Sun, 2019; Zhang et al., 2020). Such models leverage the flexibility of neural networks to estimate the intensity after each event and improve the overall model performance. In these models, the focus is on approximating a discrete set of intensities before and after each event, which are combined to interpolate the continuous intensities. For example, Du et al. (2016) uses an RNN to generate intensities after each event. Mei and Eisner (2016) proposes a novel RNN architecture that generates intensities at both ends of each inter-event interval. Furthermore, some studies have explored alternative training schema: Xiao et al. (2017) used Wasserstein distance as a training loss, Guo et al. (2018) introduced noise-contrastive estimation technique, and Li et al. (2018) leveraged reinforcement learning.

While these NPP models are more expressive than traditional ones, they still assume simple inter-event intensity changes that are continuous and monotonic. The work of Omi et al. (2019) proposes to relax this assumption by parameterizing the integral of an intensity function with a DNN and incorporating an RNN as well. However, their model can only inherit previous events' influence through an arbitrary RNN, and the learned intensity change cannot directly summate over time.

Integration Methods. Integration methods play a crucial role in a model's ability to capture the complex dynamics of a system but have largely been limited to simple techniques in NPP literature. Existing works have either used an intensity function with an elementary integral (Du et al., 2016) or relied on Monte Carlo integration (Mei and Eisner, 2016). However, we will see from our experiments that the choice of integration method has a non-trivial effect on the model performance.

Integration is generally more complicated than differentiation, as most integration rules, such as integration by parts and change of variables, transform an antiderivative to another that is not necessarily easier. While elementary antiderivatives exist for a small set of functions, they are not available for many simple composite functions like $\exp(x^2)$ (Dunham, 2018). To overcome this, the Risch algorithm was developed to determine such elementary antiderivatives (Risch, 1969, 1970). However, this algorithm has never been fully implemented due to its complexity. As a result, numerical integration methods are commonly used, such as Newton-Cotes Methods, Romberg Integration, Quadrature, and Monte Carlo integration (Davis and Rabinowitz, 2007).

Multiple recent works introduced variants of a new integration approach, Automatic Integration (AutoInt). Liu (2020) proposes integrating the Taylor polynomial using the derivatives from Automatic Differentiation (AutoDiff). It requires partitioning the integral limits and choosing the order of Taylor approximation. Though it uses the efficient AutoDiff, the integration procedure involves a trade-off between runtime and accuracy and is numerical in nature. Li et al. (2019) and Lindell et al. (2021) proposed a dual network approach which we will discuss in detail in Section 3. The method guarantees a closed-form integral and is efficient. We adapted this dual network approach to the point process settings.

3. Methodology

In this section, we first review the background of Neural Point Processes (NPP) and their limitations. Then we introduce a new NPP model, which is more interpretable and flexible. We explain how to use automatic integration with such an NPP for fast training and inference.

3.1. Point Processes and Limitations of NPPs.

Temporal Point Process. A temporal point process (TPP) is a counting process $N(t)$, representing the number of events that occurs before time t. It is characterized by a scalar non-negative *intensity function* $\lambda^*(t)$. Given the history events before time t denoted by $\mathcal{H}_t := \{t_1, ..., t_n\}_{t_n \leq t}$, the intensity function quantifies the event arrival rate at time t, and is formally defined as

$$\lambda^*(t) := \lim_{\Delta t \to 0} \frac{\mathbb{E}[N(t, t + \Delta t)|\mathcal{H}_t]}{\Delta t}.$$

The notation $*$ is from Daley and Vere-Jones (2007) to indicate the intensity is conditional on the past but not including the present. Hawkes process (Hawkes, 1971) is an example of TPP, defined as:

$$\lambda^*(t) = \mu + \alpha \sum_{t_i < t} \exp(-\beta(t - t_i)), \tag{2}$$

When a new event occurs, it produces an increment in the intensity, and this influence decays exponentially. μ is the base intensity representing the rate of an event happening on its own. α and β are scalars.

Neural Point Processes. Neural Point Process (NPP) models combine DNNs with point processes to increase their capacity. State-of-the-art NPPs first encode the events into hidden states. For temporal NPP, if the inter-event function is as simple as a scalar kernel function (Du et al., 2016), the integral is easy, but the model's expressiveness is limited. In contrast, if the inter-event function is high dimensional (Mei and Eisner, 2016; Zuo et al., 2020), the model gains stronger expressive power but requires numerical integration. For spatiotemporal NPP (Chen et al., 2020; Zhou et al., 2022), a non-negative activation function maps the hidden states to a scalar, representing the temporal intensity immediately after an event, and a conditional spatial distribution. The change of intensity between events is represented using a decay function or a Neural-ODE. The conditional spatial distribution is represented by a kernel mixture or a normalizing flow. Nevertheless, all models assume a continuous transformation of the intensity function, limiting their expressivity.

Limitations of Existing NPPs. NPPs combine DNNs with point processes to enhance their expressivity. However, they are limited when learning sophisticated intensities.

For example, RMTPP (Du et al., 2016) encodes the i-th event history as a hidden vector $\mathbf{h}(t_i^+)$. The influence of an event over time is given by the interpolation function f, such that the intensity is

$$\lambda^*(t)|_{t_i \le t \le t_{i+1}} = \mu + g^+(\mathbf{w}^T\mathbf{h}(t_i^+) + f(t - t_i)).$$

where μ is the base intensity, $\mathbf{w} \in \mathbb{R}^{1\times k}$ is a linear layer and g^+ is a positive activation function.

Neural Hawkes process (Mei and Eisner, 2016) adds a prediction of the influence right before the $i+1$-th event as the hidden state $\mathbf{h}(t_{i+1}^-)$, so that the intensity is

$$\lambda^*(t)|_{t_i \le t \le t_{i+1}} = g^+(\mathbf{w}^T f(\mathbf{h}(t_i^+), \mathbf{h}(t_{i+1}^-))).$$

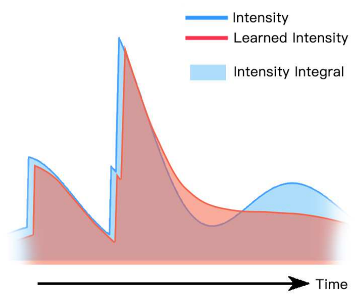

Figure 2: Failure to recover the true intensity despite the fact that estimated likelihood matches the truth (the areas under the curves are the same).

The model assumes a scalar kernel function f to linear interpolate the two hidden state vectors.

These NPPs define a separate intensity function for each interval $[t_i^+, t_{i+1}^-]$. Compared to Hawkes process in Equation 2, the use of hidden states makes it difficult to interpret the influence of each historical event. Additionally, their use of simple interpolated change of intensity in each interval cannot handle the "cyclic influence" in Figure 2. The models may yield the correct likelihood, but fail to recover the true intensity. Moreover, these models are not compatible with AutoInt. Although AutoInt can approximate any function f and its antiderivative in closed forms, finding a closed-form antiderivative for the function composition $g^+ \circ f$ is still intractable.

3.2. Influence-Driven Point Process

We propose a new NPP model that directly generalizes the Hawkes process in Equation 2. The model, driven by a complex influence function, has the following conditional intensity:

$$\lambda^*(t) = \mu + \sum_{t_i < t} f_\theta^+(t - t_i, \mathcal{H}(t_i)) := \mu + \sum_i f_\theta^+(t - t_i), f_\theta : \mathbb{R}^1 \to \mathbb{R}^1, \tag{3}$$

where μ is the scalar base intensity and f_θ^+ is a positive scalar function that takes time and event history representations $\mathcal{H}(t_i)$ as inputs.

The model has two advantages. Firstly, each f_θ^+ is approximated by a DNN, which allows for the model to capture complex inter-event intensity changes, including the "cyclic influence" scenario. Secondly, our model is more interpretable. The additive form of our model allows for interpreting different past events' influence on the current event by decomposing the intensity function.

In simplified cases, we can directly use the difference in event times, $t - t_i$ as input to f_θ^+. We consider some alternative methods for representing the event history $\mathcal{H}(\mathbf{t}_i)$ to compare with this formulation. One option is to use RNNs or Transformers to encode the accumulative influence of events as hidden vectors $\{\mathbf{h}_i\}_{i=0}^N$. The hidden vectors could be then used to scale each event's influence, resulting in a conditional intensity function formulated as

$$\lambda^*(t) = \mu + \sum_i g_\phi(\mathbf{h}_i) f_\theta^+(t - t_i), \quad f_\theta : \mathbb{R}^1 \to \mathbb{R}^1, \quad g_\phi : \mathbb{R}^1 \to \mathbb{R}^1. \tag{4}$$

Here g_ϕ is a separate neural network. Alternatively, time t and $\mathbf{h}_i$ can be concatenated and fed to the neural network, such that the conditional intensity becomes

$$\lambda^*(t) = \mu + \sum_i f_\theta^+(t - t_i \oplus \mathbf{h}_i), \quad f_\theta : \mathbb{R}^{k+1} \to \mathbb{R}^1 \tag{5}$$

However, we show in the experiments later that the introduction of recurrent vector like Omi et al. (2019) in Equations 4 and 5 is not beneficial; it increases the degree of freedom so much that it can easily overfit on real-world datasets. The use of hidden states also makes the model less interpretable.

3.3. Automatic Integration (AutoInt)

The NPPs proposed in Equation 3, 4 and 5 has a major advantage: the ability to use automatic integration (AutoInt) and calculate the integral $\int_{t=a}^{b} f_\theta(t, \mathbf{h}) := F_\theta(b, \mathbf{h}) - F_\theta(a, \mathbf{h})$ along time axis. AutoInt first constructs the integral network F_θ, and then reorganizes the computational graph of F_θ to form the integrant f_θ. The two networks thus share the same set of parameters θ.

Specifically, let $\mathbf{x} := t \oplus \mathbf{h}$, we approximate the integral of the intensity function as a DNN of the form:

$$F_\theta(\mathbf{x}) = \mathbf{W}_n ... (\mathbf{W}_3 \sigma(\mathbf{W}_2 \sigma(\mathbf{W}_1 \mathbf{x}))),$$

where $\mathbf{W}_k : \mathbb{R}^{M_k} \mapsto \mathbb{R}^{N_k}$ denotes the weight of the k-th linear layer, σ the elementwise nonlinearity, M_k and N_k the input and output dimensions of the k-th layer, and $\theta = \{\mathbf{W}_k \in \mathbb{R}^{M_k \times N_k}, \forall k\}$ the set of parameters.

The influence network f_θ is the partial derivative of the integral network F_θ. As long as the activation function is differentiable everywhere, the intensity can be computed recursively:

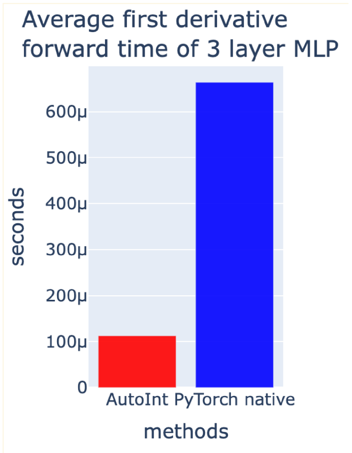

Figure 3: Comparing MLP forward time using our Integrant Net and naive PyTorch

$$f_\theta(\mathbf{x}) := \frac{\partial F_\theta}{\partial t}(\mathbf{x}) = \mathbf{W}_k \sigma'(\mathbf{W}_{k-1} \sigma(\mathbf{W}_{k-2} \ldots (\mathbf{W}_1 \mathbf{x}))) \cdots \circ \mathbf{W}_2 \sigma'(\mathbf{W}_1 \mathbf{x}) \circ \mathbf{W}_{11}$$

where $\circ$ indicates the Hadamard product, and $\mathbf{W}_{11}$ is the first column of $\mathbf{W}_1$, i.e.,

$$\mathbf{W}_1 := [\mathbf{W}_{11} \quad \mathbf{W}_{12} \quad \ldots \quad \mathbf{W}_{1,M_1}]$$

Computing $f_\theta(\mathbf{x})$ involves many repeated operations. For example, $\mathbf{W}_1\mathbf{x}$ is used for computing both $\sigma(\mathbf{W}_1\mathbf{x})$ and $\sigma'(\mathbf{W}_1\mathbf{x})$, see Figure 1. We implemented a program that leverages dynamic programming to create a derivative model efficiently using automatic differentiation, see Algorithm in A.3. We compared our implementation with PyTorch's default AutoInt and confirmed a 50% speedup on average for calculating a first derivative. With 3-layer Multi-Layer Perceptron (MLP), the advantage is up to 80%, see Figure 3.

3.4. Imposing the Non-negativity Constraint

To ensure the model in Equation 3 yields a valid intensity, we need to ensure that the function f_θ is always non-negative. Constraining all linear weights to be non-negative as in Sill (1998) is too strict since we only want the network to be monotonic for the time input and not others.

We design an AutoInt scheme tailored for NPP as shown in Figure 4: we first pass the hidden vector $\mathbf{h}$ and the time t through two linear layers with non-negative weights W^+ separately. Then, we concatenate the outputs to another non-negative weighted network. The resulting integral monotonically increases with respect to time, as the time input t does not pass through any layer with negative weights. The two unconstrained layers with weights W ensure that the expressivity of other input dimensions is not affected.

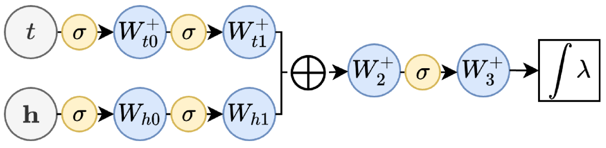

Figure 4: Monotonically increasing integral network. t is the time of the event and h is the encoded hidden vector. "W^+" indicates the neural network layer has non-negative weights.

Through experimentation, we found that projected gradient descent (Chorowski and Zurada, 2014) (i.e., clamping the weights after each optimizer step) converges better to the ground truth than the exponential transformation method. To ensure monotonicity, we use a monotonic activation function. Previous AutoInt (Lindell et al., 2021) works used the sine activation, which is non-monotonic. We found that both tanh and sine activation yield similar performance, as also indicated by Parascandolo et al. (2016)

3.5. Model Training

Given the monotonic integral network $F_\theta(t, \mathbf{h})$ and the integrant network $f_\theta = \frac{\partial F_\theta}{\partial t}$ approximating the influence function, the log-likelihood of an event sequence $\mathcal{H}_n = \{t_1, ..., t_n\}$ observed in time interval $[0, T]$ with respect to the model is

$$\mathcal{L}(\mathcal{H}_n) = \sum_{i=1}^{n} \log \left(\sum_{j=1}^{i-1} f_\theta(t_i - t_j, \mathbf{h}_i) \right) - \sum_{i=1}^{n} \Bigg(F_\theta(T - t_i, \mathbf{h}_n) - F_\theta(0, \mathbf{h}_i) \Bigg).$$

where $\{\mathbf{h}_i\}$ are the hidden states generated by a deep sequence model, but can be omitted with Equation 3. This is an application of the Fundamental Theorem of Calculus (Sobczyk and Sánchez, 2011). To learn the parameters θ in both networks, we maximize the log-likelihood function. We name our method Automatic Integration Neural Point Processes (`Auto-NPP`).

4. Experiments

We compare different NPPs using both synthetic and real-world data. Our goal for synthetic data is to validate the ability of `Auto-NPP` to accurately recover complex intensity functions. We report the predictive performance and computational cost to showcase the advantages of AutoInt.

4.1. Experimental Setup

Synthetic Datasets. We simulated three challenging synthetic point process datasets using Ogata's thinning algorithm (Chen, 2016). Each dataset contains 8192 sequences over a time range of $[0, 50)$, with a train-val-test split of $2 : 1 : 1$.

- *Shaky Hawkes process*: This dataset multiplies the influence function of the Hawkes process (see Equation 2) by a cyclic function, resulting in a multimodal intensity for long inter-event intervals. The conditional intensity function was defined as: $\lambda^*(t) = \mu + \alpha \sum_{i=1}^{N} \cos((t - t_i) + 1) \exp(-\beta(t - t_i))$. The values used for our experiments were $\alpha = \beta = \mu = 0.2$.
- *Delayed Peak process*: This dataset features a unimodal but non-smooth influence function. Starting from 0, each event's influence initially increases and then decreases, following a bell-shaped curve. The conditional intensity function was defined as: $\lambda^*(t) = \mu + \alpha \sum_{i=1}^{N} \mathrm{ReLU}(-(\beta(t - t_i) - 1)^2 + 1)$. The values used for our experiments were $\alpha = 0.2, \beta = 0.5, \mu = 0.3$.
- *Shift Hawkes process*: This dataset describes the scenario in which a post becomes viral several hours after it is first visible, resulting in a jump in the intensity between events. The conditional intensity was characterized as: $\lambda^*(t) = \mu + \alpha \sum_{i=1}^{N} \mathbf{1}(t - t_i > \gamma) \exp(-\beta(t - t_i - \gamma))$. The values used for our experiments were $\alpha = \beta = \mu = 0.2$, and the threshold $\gamma = 2.0$.

Real-world Datasets. We use two real-world benchmark datasets, *Earthquake Japan* and *COVID-19 NJ* from Chen et al. (2020). *Earthquake Japan* includes the times and locations of all earthquakes in Japan from 1990 to 2020 with magnitudes of at least 2.5. The dataset contains 1500 sequences over a time range of $[0, 30)$. The train-validation-test data split is $4 : 1 : 1$. *COVID-19 NJ* is published by The New York Times to describe the times and county locations of COVID cases in New Jersey state. The dataset contains 1650 sequences over a time range of $[0, 7)$. The train-validation-test data split is $4 : 1 : 1$. To normalize them, we scale the Earthquake dataset's times by 2 and the COVID dataset's times by 20.

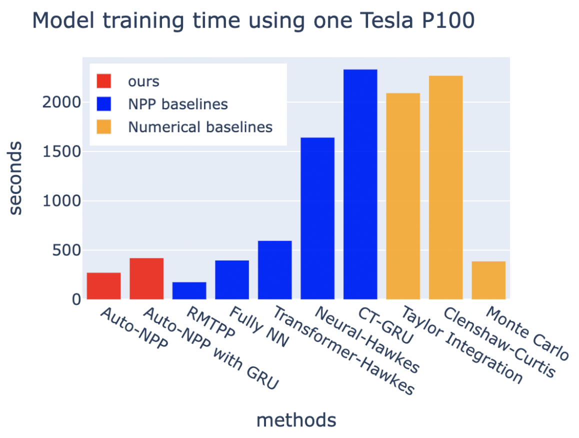

Figure 5: Training speed comparison for different NPPs and numerical integration methods in seconds. The proposed `Auto-NPP` is fast. RMTPP is the fastest but suffers from poor prediction performance.

Evaluation Metrics. As shown in Figure 2, an NPP model may yield a likelihood similar to the ground truth but still fail to learn the correct intensity. Therefore, in addition to test log-likelihood (LL), we also report the Mean Absolute Percentage Error (MAPE) of the estimated conditional intensity.

Baselines. We use two groups of baselines:

- *Numerical Integration methods*: learning the NPP model in equation 3 but with different integration techniques: Taylor integration (Liu (2020), see Appendix), the Clenshaw–Curtis quadrature, the Monte Carlo integration, and AutoInt.

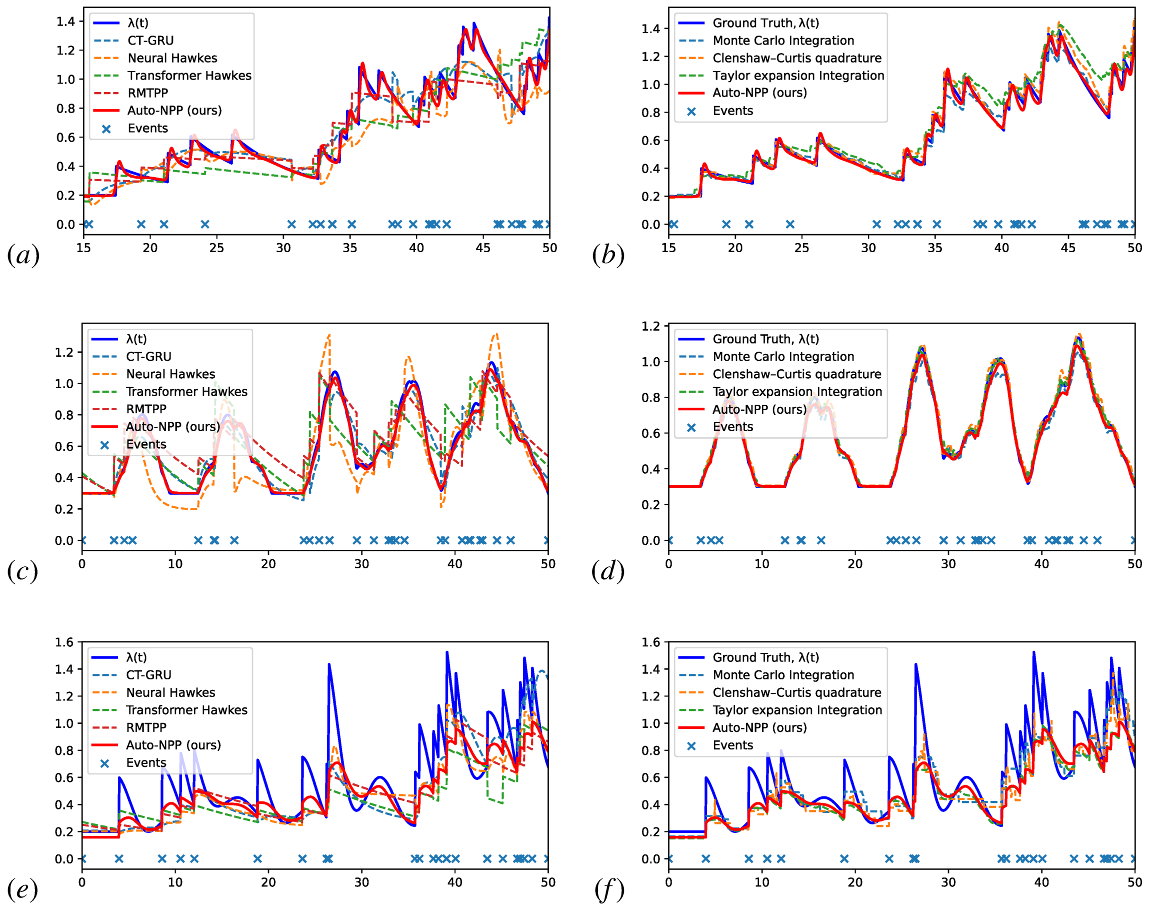

Figure 6: Visualizations of the true conditional intensity $\lambda^*(t)$ and the learned conditional intensity on the *Shift Hawkes* (a, b), *Delayed Peak* (c, d), *Shaky Hawkes* (e, f) datasets. (a, c, e): comparison of intensities learned with different models. (b, d, f): comparison of intensities learned with different integration methods.

- *Neural Point Processes*: state-of-the-art NPP models (first section), including RMTPP (Du et al., 2016), Neural-Hawkes (Mei and Eisner, 2016) and Transformer Hawkes (Zuo et al., 2020). Additionally, Mozer et al. (2017) proposed a continuous-time GRU that interpolates hidden states between events. It has a similar idea to Neural-Hawkes's continuous-time LSTM. We include a CT-GRU variant of Neural-Hawkes to increase the diversity of our baselines.

4.2. Experimental Results

Table 1 compares the prediction Mean Absolute Percentage Error (MAPE) and test log-likelihood (LL) between `Auto-NPP` and the baseline models on the synthetic datasets. `Auto-NPP` has a significant advantage on the synthetic dataset with complex intensity. Figure 6 further compares the ground truth intensity and the learned intensities. While using numerical integration like Monte Carlo may occasionally yield a higher likelihood estimate, AutoInt most effectively recovers the ground truth intensity as shown by the MAPE. Moreover, we can see that our method is the only one that can capture the multimodal intensity function, as shown in Figure 6(e, f). The bias makes numerical methods more likely to learn flatter intensity.

Model	shakyHawkes		shiftHawkes		decayPeak	
	MAPE	LL	MAPE	LL	MAPE	LL
CT-GRU (Mozer et al., 2017)	0.2243	-35.6063	0.1262	-39.7173	0.1103	-42.1959
Neural Hawkes (Mei and Eisner, 2016)	0.2168	-35.4043	0.1473	-40.0411	0.1468	-42.5548
RMTPP (Du et al., 2016)	0.2562	-35.6549	0.2630	-39.7893	0.2183	-42.7965
Transformer Hawkes (Zuo et al., 2020)	0.2812	-36.1831	0.2316	-40.6717	0.2342	-43.3308
Fully Hawkes (Omi et al., 2019)	0.6435	-64.7072	0.4522	-58.4046	0.4100	-65.0632
Clenshaw-Curtis	0.2197	-35.5183	0.0541	-39.4831	0.0312	-41.9839
Monte Carlo	0.1935	-35.6090	0.0462	**-39.3527**	0.0378	-41.9868
Taylor Expansion (Liu, 2020)	0.2004	-35.3771	0.0999	-39.7062	**0.0224**	-41.9691
`Auto-NPP`	**0.1843**	**-35.3762**	**0.0356**	-39.3599	0.0226	**-41.9678**
`Auto-NPP` (w/ RNN)	0.3353	-37.9182	0.4675	-44.0076	0.1107	-42.3124

Table 1: Comparison between our proposed model `Auto-NPP` (with or without RNN), the state-of-the-art NPP models (upper section), and the same model using different integration methods (lower section) on three synthetic datasets. Performance w.r.t. Mean Absolute Percentange Error (MAPE) of the estimated intensity $\lambda^*(t)$ and Test log likelihood (LL).

Table 2 compares the test log-likelihood (LL) on the real-world datasets. Our method outperforms most other state-of-the-art methods for forecasting COVID events and earthquakes. These results demonstrate the model's capability to learn complex real-world dynamics. Conversely, the poor performance of the RNN variant of `Auto-NPP` and Fully RNN reveal that combining an RNN with an unconstrained influence function can overfit.

Finally, Figure 5 compares the training time of different methods. Auto-NPP is faster than most state-of-the-art NPP models, and AutoInt is not only more accurate but also faster than all other integration methods.

5. Conclusion

We propose Automatic Integration for Neural point process models (`Auto-NPP`) using a dual network approach. `Auto-NPP` can efficiently compute the exact likelihood of *any* sophisticated intensity. We validate our approach using both synthetic point processes with complex intensity functions and real-world datasets. Experiment results demonstrate that `Auto-NPP` can accurately recover the underlying intensity function while being efficient.

Our work presents a new paradigm for learning discrete event data with continuous-time dynamics. Presently, our neural process model solely takes the form of Hawkes processes that exhibit self-exciting behavior. However, it cannot handle self-correcting processes owing to the complexity of integration. In future work, we aim to relax the form of the intensity network leveraging advanced integration techniques.

Model	earthquakesJP	covidNJ
	LL	LL
CT-GRU	-37.7458	-24.6818
Neural Hawkes	**-36.3709**	-25.4887
RMTPP	-39.0440	-22.6118
Transformer Hawkes	-41.3816	-23.2670
Fully Hawkes	-55.2647	-65.0208
Clenshaw-Curtis	-38.7512	-21.9713
Monte Carlo	-38.7560	-22.0792
Taylor Expansion	-38.5432	-22.0331
`Auto-NPP`	-38.4888	**-21.8996**
`Auto-NPP` (w/ RNN)	-39.1172	-23.8340

Table 2: Comparison between the models and the integration methods on two real world datasets, earthquakesJP and covidNJ. Performance w.r.t. only LL since there is no ground truth intensity.

Acknowledgments

This work was supported in part by U.S. Department Of Energy, Office of Science, Facebook Data Science Research Awards, U. S. Army Research Office under Grant W911NF-20-1-0334, and NSF Grants #2134274 and #2146343.

References

Ryan Prescott Adams, Iain Murray, and David JC MacKay. Tractable nonparametric bayesian inference in poisson processes with gaussian process intensities. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 9–16, 2009.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 6572–6583, 2018.

Ricky TQ Chen, Brandon Amos, and Maximilian Nickel. Neural spatio-temporal point processes. *arXiv preprint arXiv:2011.04583*, 2020.

Yuanda Chen. Thinning algorithms for simulating point processes. *Florida State University, Tallahassee, FL*, 2016.

Jan Chorowski and Jacek M Zurada. Learning understandable neural networks with nonnegative weight constraints. *IEEE transactions on neural networks and learning systems*, 26(1):62–69, 2014.

John P Cunningham, Krishna V Shenoy, and Maneesh Sahani. Fast gaussian process methods for point process intensity estimation. In *Proceedings of the 25th international conference on Machine learning*, pages 192–199, 2008.

Daryl J Daley and David Vere-Jones. *An introduction to the theory of point processes: volume II: general theory and structure*. Springer Science & Business Media, 2007.

Philip J Davis and Philip Rabinowitz. *Methods of numerical integration*. Courier Corporation, 2007.

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 and data mining*, pages 1555–1564, 2016.

William Dunham. *The calculus gallery*. Princeton University Press, 2018.

Ruocheng Guo, Jundong Li, and Huan Liu. Initiator: Noise-contrastive estimation for marked temporal point process. In *IJCAI*, pages 2191–2197, 2018.

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

Hengguan Huang, Hao Wang, and Brian Mak. Recurrent poisson process unit for speech recognition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6538–6545, 2019.

Athanasios Kottas and Bruno Sansó. Bayesian mixture modeling for spatial poisson process intensities, with applications to extreme value analysis. *Journal of Statistical Planning and Inference*, 137(10):3151–3163, 2007.

Haibin Li, Yangtian Li, and Shangjie Li. Dual neural network method for solving multiple definite integrals. *Neural computation*, 31(1):208–232, 2019.

Shuang Li, Shuai Xiao, Shixiang Zhu, Nan Du, Yao Xie, and Le Song. Learning temporal point processes via reinforcement learning. *arXiv preprint arXiv:1811.05016*, 2018.

David B Lindell, Julien NP Martel, and Gordon Wetzstein. Autoint: Automatic integration for fast neural volume rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14556–14565, 2021.

Keqin Liu. Automatic integration. *arXiv e-prints*, pages arXiv–2006, 2020.

Hongyuan Mei and Jason Eisner. The neural hawkes process: A neurally self-modulating multivariate point process. *arXiv preprint arXiv:1612.09328*, 2016.

Jesper Møller, Anne Randi Syversveen, and Rasmus Plenge Waagepetersen. Log gaussian cox processes. *Scandinavian journal of statistics*, 25(3):451–482, 1998.

Michael C Mozer, Denis Kazakov, and Robert V Lindsey. Discrete event, continuous time rnns. *arXiv preprint arXiv:1710.04110*, 2017.

Takahiro Omi, Naonori Ueda, and Kazuyuki Aihara. Fully neural network based model for general temporal point processes. *arXiv preprint arXiv:1905.09690*, 2019.

Giambattista Parascandolo, Heikki Huttunen, and Tuomas Virtanen. Taming the waves: sine as activation function in deep neural networks. 2016.

Stephen L Rathbun and Noel Cressie. Asymptotic properties of estimators for the parameters of spatial inhomogeneous poisson point processes. *Advances in Applied Probability*, 26(1):122–154, 1994.

Robert H Risch. The problem of integration in finite terms. *Transactions of the American Mathematical Society*, 139:167–189, 1969.

Robert H Risch. The solution of the problem of integration in finite terms. *Bulletin of the American Mathematical Society*, 76(3):605–608, 1970.

Jin Shang and Mingxuan Sun. Geometric hawkes processes with graph convolutional recurrent neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4878–4885, 2019.

Oleksandr Shchur, Ali Caner Türkmen, Tim Januschowski, and Stephan Günnemann. Neural temporal point processes: A review. *arXiv preprint arXiv:2104.03528*, 2021.

Joseph Sill. Monotonic networks. 1998.

Garret Sobczyk and Omar León Sánchez. Fundamental theorem of calculus. *Advances in Applied Clifford Algebras*, 21:221–231, 2011.

Utkarsh Upadhyay, Abir De, and Manuel Gomez-Rodriguez. Deep reinforcement learning of marked temporal point processes. *arXiv preprint arXiv:1805.09360*, 2018.

Shuai Xiao, Mehrdad Farajtabar, Xiaojing Ye, Junchi Yan, Le Song, and Hongyuan Zha. Wasserstein learning of deep generative point process models. *arXiv preprint arXiv:1705.08051*, 2017.

Junchi Yan, Xin Liu, Liangliang Shi, Changsheng Li, and Hongyuan Zha. Improving maximum likelihood estimation of temporal point process via discriminative and adversarial learning. In *IJCAI*, pages 2948–2954, 2018.

Qiang Zhang, Aldo Lipani, Omer Kirnap, and Emine Yilmaz. Self-attentive hawkes process. In *International Conference on Machine Learning*, pages 11183–11193. PMLR, 2020.

Zihao Zhou, Xingyi Yang, Ryan Rossi, Handong Zhao, and Rose Yu. Neural point process for learning spatiotemporal event dynamics. In *Learning for Dynamics and Control Conference*, pages 777–789. PMLR, 2022.

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.