UC San Diego

UC San Diego Electronic Theses and Dissertations

Title

Neural Point Process for Learning Spatiotemporal Event Dynamics

Permalink

https://escholarship.org/uc/item/3sn484cn

Author

Zhou, Zihao

Publication Date

2022

Peer reviewed|Thesis/dissertation

eScholarship.org Powered by the California Digital Library
University of California

UNIVERSITY OF CALIFORNIA SAN DIEGO

Neural Point Process for Learning Spatiotemporal Event Dynamics

A thesis submitted in partial satisfaction of the requirements
for the degree Master of Science

in

Computer Science

by

Zihao Zhou

Committee in charge:

Professor Rose Yu, Chair
Professor Sicun Gao
Professor Lawrence Saul

2022

Copyright
Zihao Zhou, 2022
All rights reserved.

The thesis of Zihao Zhou is approved, and it is acceptable in quality and form for publication on microfilm and electronically.

University of California San Diego

2022

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I would like to acknowledge Professor Rose Yu for her support as the chair of my committee. Through multiple drafts and many long nights, her guidance has proved to be invaluable.

I would like to thank Xingyi Yang for contributing to the exploratory experiments. His inspiration and professional attitude are appreciable.

This work was supported by Adobe Research. I would also like to give special thanks to Dr. Ryan A. Rossi and Dr. Handong Zhao for their advice in improving the thesis.

Last but not least, I would like to thank my parents—Saijun Zhou and Qiuhong Xiong, for giving birth to me and supporting me throughout my life.

ABSTRACT OF THE THESIS

Neural Point Process for Learning Spatiotemporal Event Dynamics

by

Zihao Zhou

Master of Science in Computer Science

University of California San Diego, 2022

Professor Rose Yu, Chair

Learning the dynamics of spatiotemporal events is a fundamental problem. Neural point processes enhance the expressivity of point process models with deep neural networks. However, we notice two limitations of most existing methods: they only consider temporal dynamics without spatial modeling, and they apply numerical integration that may suffer additional numerical errors and high computational costs. This thesis explores new options in modeling neural point processes and proposes two models. The first model, Deep Spatiotemporal Point Process (`DeepSTPP`), constrains the intensity function to an integrable functional form. We assume the intensity function to be governed by a latent process to improve the expressivity of the model. We use amortized variational inference to infer the latent process with deep networks. The second model, *Automatic*

Integration for Neural point process models (`AIN`), further improves the expressivity. It relies on a dual network approach that learns the intensity and its integral together in closed forms. As it does not pose any assumption over the intensity function form, it can learn irregular time series data governed by complex intensity functions. Both models are flexible, efficient, and can accurately learn dynamics behind irregularly sampled events over space and time. Using synthetic datasets, we validate our models can accurately learn the true intensity function. On real-world benchmark datasets, our models demonstrate superior performance over state-of-the-art baselines.

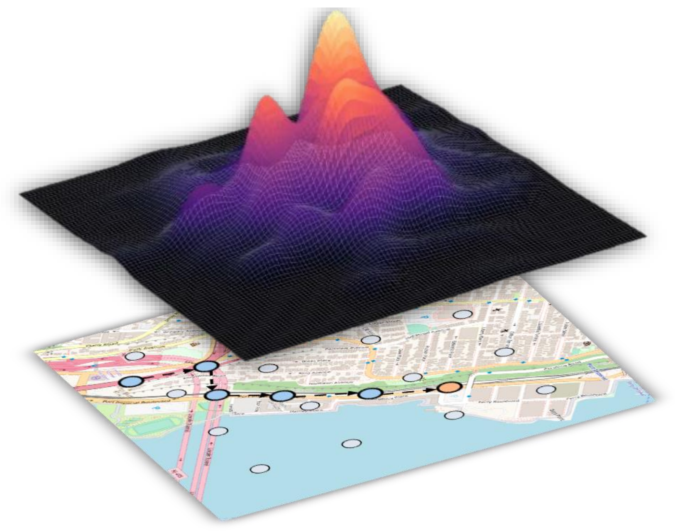
Introduction

Accurate modeling of spatiotemporal event dynamics is fundamentally important for disaster response (Veen and Schoenberg, 2008), logistic optimization (Safikhani et al., 2018) and social media analysis (Liang et al., 2019). Compared to other sequence data such as texts or time series, spatiotemporal events occur irregularly with uneven time and space intervals.

Discrete-time deep dynamics models such as recurrent neural networks (RNNs) (Hochreiter and Schmidhuber, 1997; Chung et al., 2014) assume events to be evenly sampled. Interpolating an irregular sampled sequence into a regular sequence can introduce significant biases (Rehfeld et al., 2011). Furthermore, event sequences contain strong spatiotemporal dependencies. The rate of an event depends on the preceding events, as well as the events geographically correlated to it.

Spatiotemporal point processes (STPP) (Daley and Vere-Jones, 2007; Reinhart, 2018) provides the statistical framework for modeling continuous-time event dynamics. As shown in Figure 0.1, given the history of events sequence, STPP estimates the intensity function that is evolving in space and time. However, traditional statistical methods for estimating STPPs often require strong modeling assumptions, feature engineering, and can be computationally expensive.

Machine learning community is observing a growing interest in continuous-time deep dynamics models that can handle irregular time intervals. For example, Neural ODE (Chen et al., 2018)



Historical events $\{(\mathbf{s}_{i-j}, t_{i-j})\}_{j=1}^{J}$
Upcoming event $(\mathbf{s}_i, t_i)$
User's trajectory
Representive points

Figure 0.1: Illustration of learning spatiotemporal point process. We aim to learn the space-time intensity function given the historical event sequence and representative points as background.

parametrizes the hidden states in an RNN with an ODE. Shukla and Marlin (2018) uses a separate network to interpolates between reference time points. Neural temporal point process (TPP) (Mei and Eisner, 2017; Zhang et al., 2020b; Zuo et al., 2020) is an exciting area that combines fundamental concepts from temporal point processes with deep learning to model continuous-time event sequences, see a recent review on neural TPP (Shchur et al., 2021). However, most of the existing models only focus on *temporal* dynamics without considering *spatial* modeling.

In the real world, while time is a unidirectional process (arrow of time), space extends in multiple directions. This fundamental difference from TPP makes it nontrivial to design a unified STPP model. The naive approach to approximate the intensity function by a deep neural network would lead to intractable integral computation for likelihood. Unlike marked TPPs or RNNs with spatiotemporal output, neural STPP models exploit the continuous nature of the spatiotemporal domain and provide an interpretable scalar latent dynamic. One can see in neural STPP models how each event affects the arrival rate of future events at different locations, while marked TPPs or RNNs can only output a spatial location from a black box.

Prior research such as Du et al. (2016) discretizes the space as "markers" and use marked TPP to classify the events. This approach cannot produce the space-time intensity function. Okawa et al. (2019) models the spatiotemporal density using a mixture of symmetric kernels, which ignores the unidirectional property of time. Chen et al. (2020) proposes to model temporal intensity and spatial density separately with neural ODE, which is computational expensive.

The main challenge in extending a TPP framework to an STPP framework is the multivariate integration of the intensity function when calculating the likelihood. Suppose we directly use a neural network with scalar output to model the intensity function. If the intensity is univariate, we can use the Monte Carlo method to estimate its integral efficiently. But when it comes to the high-dimensional spatiotemporal domain, the number of data points required for accurate estimation grows exponentially. Alternatively, if we have a prior assumption about the intensity function like it follows an exponential decay at a fixed rate, we may severely weaken the expressivity of the model.

Our contribution is developing efficient neural point processes that effectively capture the latent continuous-time dynamic behind discrete spatiotemporal events. We tackle the above challenges with two different models. In Chapter 2, we represent the intensity using integrable kernel function. Nevertheless, we parametrize the kernel using a latent process to give it better expressive power. The latent process consists of deep neural networks and is learned using variational inference. In Chapter 3, we apply the automatic integration technique to represent the intensity as a constrained integrable neural network. We demonstrate the drawbacks of learning intensity using numerical integration and the need to assume the same influence function to control the degree of freedom.

We verify the effectiveness of both models by experiments on synthetic and real-world datasets.

Chapter 1

Background

1.1 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, $\mathcal{H}_t := \{t_1, ..., t_n\}_{t_n \le t}$, the intensity function quantifies the event arrival rate at t, and is formally defined as

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

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

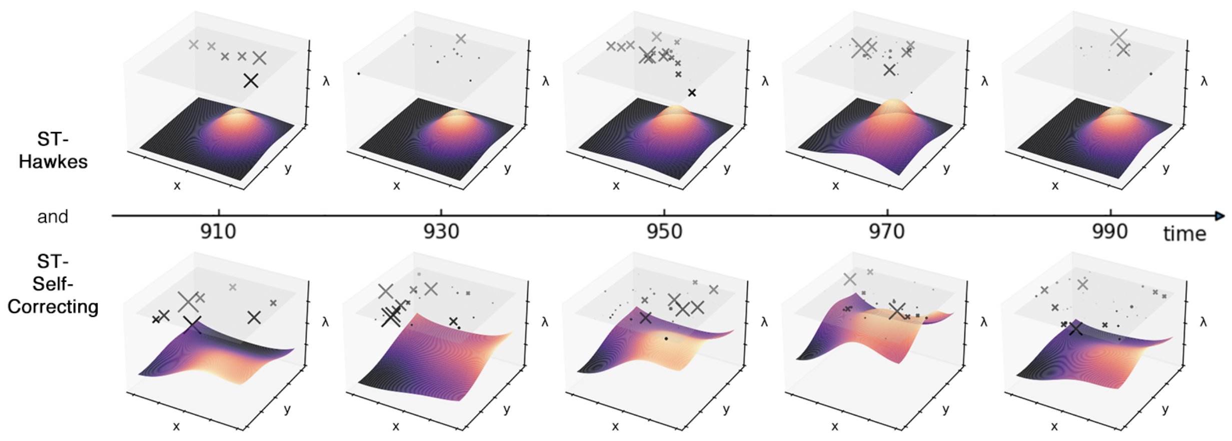


Figure 1.1: Visualization of the two example spatiotemporal point processes.

characterized by

$$\lambda^*(t) = \mu + \alpha \sum_{t_i<t} \exp(-\beta(t - t_i)), \tag{1.1}$$

where μ, α, β are scalars. The arrival of a new event results in a sudden increase of intensity, and the influence of this event will decay exponentially. μ is the base intensity representing the rate of an event happening on its own.

1.2 Spatiotemporal Point Process.

Spatiotemporal point process (STPP) models the number of events $N(\mathcal{S} \times (a, b))$ that occurred in the Cartesian product of the spatial domain $\mathcal{S} \subseteq \mathbb{R}^2$ and the time interval $(a, b]$. It is characterized by a non-negative *space-time intensity function* given the history $\mathcal{H}_t := \{(\mathbf{s}_1, t_1), \ldots, (\mathbf{s}_n, t_n)\}_{t_n \leq t}$:

$$\lambda^*(\mathbf{s}, t) := \lim_{\Delta\mathbf{s}\to 0, \Delta t\to 0} \frac{\mathbb{E}[N(B(\mathbf{s}, \Delta\mathbf{s}) \times (t, t + \Delta t))|\mathcal{H}_t]}{B(\mathbf{s}, \Delta\mathbf{s})\Delta t} \tag{1.2}$$

which is the probability of finding an event in an infinitesimal time interval $(t, t + \Delta t]$ and an infinitesimal spatial ball $\mathcal{S} = B(\mathrm{s}, \Delta\mathrm{s})$ centered at location s.

Example 1: Spatiotemporal Hawkes process (STH). Spatiotemporal Hawkes (or self-exciting) process assumes every past event has an additive, positive, decaying, and spatially local influence over future events. Such a pattern resembles neuronal firing and earthquakes. It is characterized by the following intensity function (Reinhart, 2018):

$$\lambda^*(\mathrm{s}, t) := \mu g_0(\mathrm{s}) + \sum_{i:t_i<t} g_1(t, t_i) g_2(\mathrm{s}, \mathrm{s}_i) : \mu > 0 \tag{1.3}$$

where $g_0(\mathrm{s})$ is the probability density of a distribution over $\mathcal{S}$, g_1 is the triggering kernel and is often implemented as the exponential decay function, $g_1(\Delta t) := \alpha \exp(-\beta \Delta t) : \alpha, \beta > 0$, and $g_2(\mathrm{s}, \mathrm{s}_i)$ is the density of an unimodal distribution over $\mathcal{S}$ centered at s_i.

Example 2: Spatiotemporal Self-Correcting process (STSC). Self-correcting spatiotemporal point process (Isham and Westcott, 1979) assumes that the background intensity increases with a varying speed at different locations, and the arrival of each event reduces the intensity nearby. STSC can model certain regular event sequences, such as an alternating home-to-work travel sequence. It has the following intensity function:

$$\lambda^*(\mathrm{s}, t) = \mu \exp\left(g_0(\mathrm{s})\beta t - \sum_{i:t_i<t} \alpha g_2(\mathrm{s}, \mathrm{s}_i)\right) : \alpha, \beta, \mu > 0 \tag{1.4}$$

Here $g_0(\mathrm{s})$ is the density of a distribution over $\mathcal{S}$, and $g_2(\mathrm{s}, \mathrm{s}_i)$ is the density of an unimodal distribution over $\mathcal{S}$ centered at location s_i.

Figure 1.1 illustrates the intensity functions of the two example STPPs. One can see how new events lift the neighborhood intensity for STH and suppress the neighborhood intensity for STSC.

1.3 Neural Point Process

Neural Point Process (NPP) models (Mei and Eisner, 2016; Du et al., 2016; Zuo et al., 2020) combine deep neural networks with TPPs. State-of-the-art NPPs first encode the events into hidden representations using either Recurrent Neural Network (RNN) or Transformer. Then they use a non-negative activation function to map the hidden vectors to a scalar, i.e., the intensity immediately after an event. The change of intensity between events is usually represented using a linear or exponential decay function. (Mei and Eisner, 2016) allow this decay occur in a high-dimensional space before mapping to a scalar.

By parametrizing the intensity function as $\lambda^*_\theta(t)$, the log likelihood of an event sequence $\{t_1, ..., t_N\}$ observed in time interval $[0, T]$ is

$$\mathcal{L}(\theta|\{t_1, ..., t_N\}) = \sum_{i=1}^{N} \log \lambda^*_\theta(t_i^-) - \int_{t=0}^{T} \lambda^*_\theta(t).$$

As the intensity is discontinuous at every event arrival t_i, t_i^- denotes the intensity immediately before the i-th event. The second term is usually evaluated separately for each inter-event interval (t_i, t_{i+1}). If the inter-event function is as simple as a scalar kernel function (Du et al., 2016), then the integral is easy, but the model is less expressive. On the other hand, if the inter-event function is high dimensional (Mei and Eisner, 2016; Zuo et al., 2020), then the model gains stronger expressive power at the cost of requiring numerical integration. We found in our experiments that the numerical integration errors may prevent the model from recovering the true underlying inten-

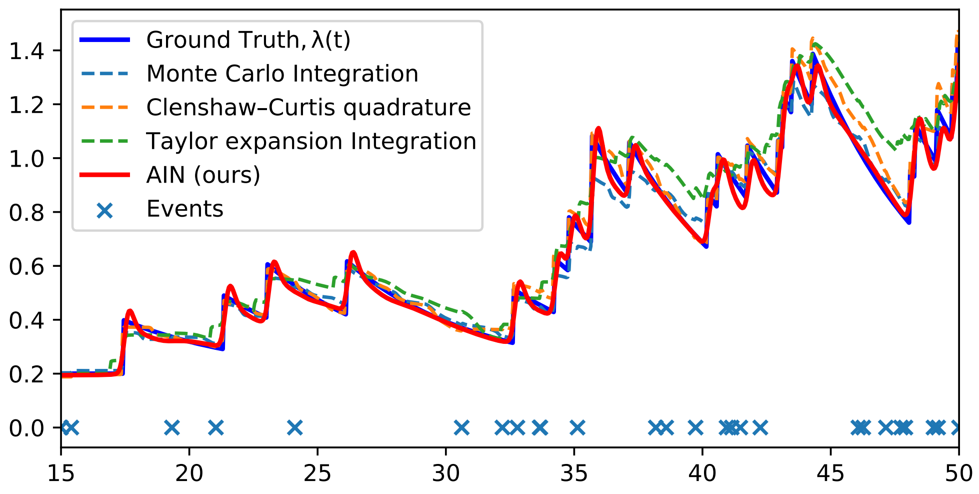


Figure 1.2: Intensities learned using different numerical integration methods, compared to our approach (`AIN`) with AutoInt. Blue crosses represent event over time. Numerical integration error can prevent the model from learning the truth intensity.

sity, see Figure 1.2. Nevertheless, all models assume a continuous transformation of the intensity function and have limited expressivity.

1.4 Maximum likelihood Estimation

Given a history of n events $\mathcal{H}_t$, the joint log-likelihood function of the observed events for STPP is as follows:

$$\log p(\mathcal{H}_t) = \sum_{i=1}^{n} \log \lambda^*(\mathbf{s}_i, t_i) - \int_{\mathcal{S}} \int_0^t \lambda^*(\mathbf{u}, \tau) d\mathbf{u} d\tau \tag{1.5}$$

Here, the space-time intensity function $\lambda^*(\mathrm{s}, t)$ plays a central role. Maximum likelihood estimation seeks the optimal $\lambda^*(\mathrm{s}, t)$ from data that optimizes Eqn. (1.5).

1.5 Predictive distribution

Denote the probability density function (PDF) for STPP as $f(\mathrm{s}, t|\mathcal{H}_t)$ which represents the conditional probability that next event will occur at location s and time t, given the history. The PDF is closely related to the intensity function:

$$f(\mathbf{s}, t|\mathcal{H}_t) \quad = \frac{\lambda^*(\mathbf{s}, t)}{1 - F^*(\mathbf{s}, t|\mathcal{H}_t)} = \lambda^*(\mathbf{s}, t) \exp\left(-\textstyle\int_{\mathcal{S}} \int_{t_n}^{t} \lambda^*(\mathbf{u}, \tau) d\tau d\mathbf{u}\right) \tag{1.6}$$

where F is the cumulative distribution function (CDF), see derivations in Appendix A.1. This means the intensity function specifies the expected number of events in a region conditional on the past.

The predicted time of the next event is the expected value of the predictive distribution for time $f^{\star}(t)$ in the entire spatial domain:

$$\mathbb{E}[t_{n+1}|\mathcal{H}_t] = \int_{t_n}^{\infty} t \int_{\mathcal{S}} f^*(\mathbf{s}, t) d\mathbf{s} dt = \int_{t_n}^{\infty} t \exp\left(-\int_{t_n}^{t} \lambda^*(\tau) d\tau\right) \lambda^*(t) d\mathbf{s} dt$$

Similarly, the predicted location of the next event evaluates to:

$$\mathbb{E}[\mathbf{s}_{n+1}|\mathcal{H}_t] = \int_{\mathcal{S}} \mathbf{s} \int_{t_n}^{\infty} f^*(\mathbf{s}, t) dt d\mathbf{s} = \int_{t_n}^{\infty} \exp\left(-\int_{t_n}^{t} \lambda^*(\tau) d\tau\right) \int_{\mathcal{S}} \mathbf{s} \lambda^*(\mathbf{s}, t) d\mathbf{s} dt$$

Chapter 2

Deep Spatiotemporal Point Process

Unfortunately, Eqn. (1.5) is generally intractable. It requires either strong modeling assumptions or expensive Monte Carlo sampling. We propose a simple yet efficient approach to learn STPP. Our model, *Deep Spatiotemporal Point Process* (`DeepSTPP`) marries the principles of spatiotemporal point processes with deep learning. We take a non-parametric approach and model the space-time intensity function as mixture of kernels. The parameters of the intensity function are governed by a latent stochastic process which captures the uncertainty of the event sequence. The latent process is then inferred via amortized variational inference. That is, we draw a sample from the variational distribution for every event. We use a Transformer network to parametrize the variational distribution conditioned on the previous events.

Compared with existing approaches, our model is non-parametric, hence does not make assumptions on the parametric form of the distribution. Our approach learns the space-time intensity function jointly without requiring separate models for time-intensity function and spatial density as in Chen et al. (2020). Our model is probabilistic by nature and can describe various uncertainties in the data. More importantly, our model enjoys closed form integration, making it feasible for processing large-scale event datasets. To summarize, our work makes the following key contributions:

- **Deep Spatiotemporal Point Process.** We propose a novel Deep Point Process model for forecasting unevenly sampled spatiotemporal events. It integrates deep learning with spatiotemporal point processes to learn continuous space-time dynamics.
- **Neural Latent Process.** We model the space-time intensity function using a nonparametric approach, governed by a latent stochastic process. We use amortized variational inference to perform inference on the latent process conditioned on the previous events.
- **Effectiveness.** We demonstrate our model using many synthetic and real-world spatiotemporal event forecasting tasks, where it achieves superior performance in accuracy and efficiency. We also derive and implement efficient algorithms for simulating STPPs.

2.1 Methodology

Our model (1) introduces a latent process to capture the uncertainty (2) parametrizes the latent process with deep neural networks to increase model expressivity and (3) approximates the intensity function with a set of spatial and temporal kernel functions.

2.1.1 Neural latent process.

Given a sequence of n event, we wish to model the conditional density of observing the next event given the history $f(\mathrm{s}, t|\mathcal{H}_t)$. We introduce a latent process to capture the uncertainty of the

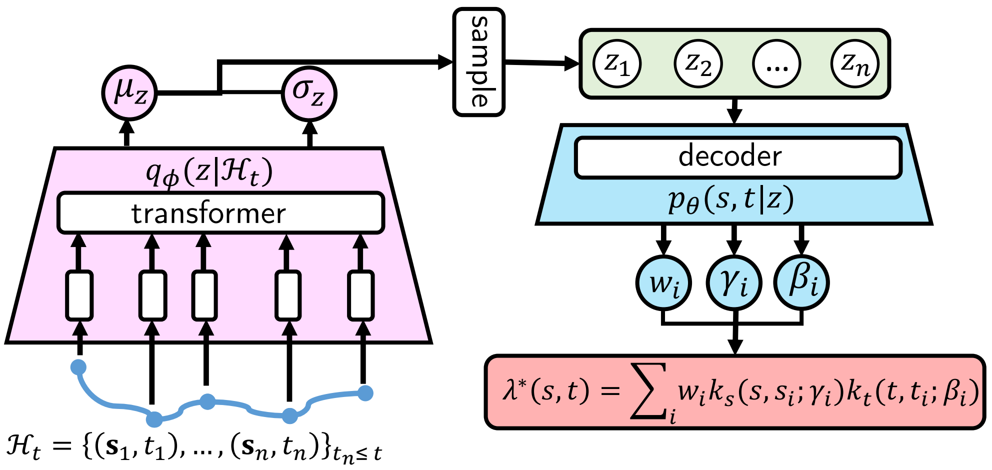

sample
z_1 z_2 ... z_n
decoder
$p_\theta(s,t|z)$
μ_z
σ_z
$q_\phi(z|\mathcal{H}_t)$
transformer
w_i γ_i β_i
$\lambda^*(s,t) = \sum_i w_i k_s(s,s_i;\gamma_i) k_t(t,t_i;\beta_i)$
$\mathcal{H}_t = \{(\mathbf{s}_1,t_1),\dots,(\mathbf{s}_n,t_n)\}_{t_n \le t}$

Figure 2.1: Design of our `DeepSTPP` model. For an event sequence, we encode it with a transformer network and map to the latent process $(z_1, \cdots, z_n)$. We use a decoder to generate the parameters (w_i, γ_i, β_i) for each event i given the latent process. The estimate intensity is calculated using the decoded parameters.

event history and infer the latent process with armotized variational inference. The latent process dictates the parameters in the space-time intensity function. We sample from the latent process using the re-parameterization trick (Kingma and Welling, 2013).

As shown in Figure 2.1, given the event sequence $\mathcal{H}_t = \{(\mathbf{s}_1, t_1), \dots, (\mathbf{s}_n, t_n)\}_{t_n \le t}$, we encode the entire sequence into the high-dimensional embedding. We use positional encoding to encode the sequence order. To capture the stochasticity in the temporal dynamics, we introduce a latent process $z = (z_1, \cdots, z_n)$ for the entire sequence. We assume the latent process follows a multivariate Gaussian at each time step:

$$z_i \sim q_\phi(z_i|\mathcal{H}_t) = \mathcal{N}(\mu, \mathrm{Diag}(\sigma)) \tag{2.1}$$

where the mean μ and covariance $\mathrm{Diag}(\sigma)$ are the outputs of the embedding neural network. In our implementation, we found using a Transformer (Vaswani et al., 2017) with sinusoidal positional encoding to be beneficial. The positions to be encoded are the normalized event time instead of the index number, to account for the unequal time interval. Recently, Zuo et al. (2020) also demonstrated that Transformer enjoys better performance for learning the intensity in temporal point processes.

2.1.2 Non-parametric model.

We take a non-parameteric approach to model the space-time intensity function $\lambda^*(\mathbf{s}, t)$ as:

$$\lambda^*(\mathbf{s}, t|z) = \sum_{i=1}^{n+J} w_i k_s(\mathbf{s}, \mathbf{s}_i; \gamma_i) k_t(t, t_i; \beta_i) \tag{2.2}$$

Here $w_i(z), \gamma_i(z), \beta_i(z)$ are the parameters for each event that is conditioned on the latent process. Specifically, w_i represents the non-negative intensity magnitude, implemented with a soft-plus activation function. $k_s(\cdot, \cdot)$ and $k_t(\cdot, \cdot)$ are the spatial and temporal kernel functions, respectively. For both kernel functions, we parametrize them as a normalized RBF kernel:

$$k_s(\mathbf{s}, \mathbf{s}_i) = \alpha^{-1} \exp\big(-\gamma_i \|\mathbf{s} - \mathbf{s}_i\|\big), \quad k_t(t, t_i) = \exp\big(-\beta_i \|t - t_i\|\big) \tag{2.3}$$

where the bandwidth parameter γ_i controls an event's influence over the spatial domain. The parameter β_i is the decay rate that represents the event's influence over time. $\alpha = \int_{\mathcal{S}} \exp\big(-\gamma_i \|\mathbf{s} - \mathbf{s}_i\|\big) d\mathbf{s}$ is the normalization constant.

We use a decoder network to generate the parameters $\{w_i, \gamma_i, \beta_i\}$ given z separately, shown in Figure 2.1. Each decoder is a 4-layer feed-forward network. We use a softplus activation function to ensure w_i and γ_i are positive. The decay rate β_i can be any number, such that an event could have constant or increasing triggering intensity over time.

In addition to n historical events, we also randomly sample J representative points from the spatial domain to approximate the background intensity. This is to account for the influence from unobserved events in the background, with varying rates at different absolution locations. The inclusion of these representative points can approximate this background distribution.

The model design in (2.2) enjoys a closed form integration, which gives the conditional PDF as:

$$f(\mathbf{s}, t|\mathcal{H}_t, z) = \lambda^*(\mathbf{s}, t|z) \exp\left(-\sum_{i=1}^{n+J} \frac{w_i}{\beta_i}[k_t(t_n, t_i) - k_t(t, t_i)]\right) \tag{2.4}$$

See the derivation details in Appendix A.2. `DeepSTPP` circumvents the integration of the intensity function and enjoys fast inference in forecasting future events. In contrast, NSTPP (Chen et al., 2020) is relatively inefficient as its ODE solver also requires additional numerical integration.

2.1.3 Parameter learning.

Due to the latent process, the posterior becomes intractable. Instead, we use amortized inference by optimizing the evidence lower bound (ELBO) of the likelihood. In particular, given event history $\mathcal{H}_t$, the conditional log-likelihood of the next event is:

$$\log p(\mathbf{s}, t|\mathcal{H}_t) \geq \log p_\theta(\mathbf{s}, t|\mathcal{H}_t, z) + \text{KL}(q_\phi(z|\mathcal{H}_t)||p(z)) \tag{2.5}$$

$$= \log \lambda^*(\mathbf{s}, t|z) - \int_{t_n}^{t} \lambda^*(\tau)d\tau + \text{KL}(q||p) \tag{2.6}$$

where ϕ represents the parameters of the encoder network and θ are the parameters of the decoder network. $p(z)$ is the prior distribution, which we assume to be Gaussian. $\text{KL}(\cdot||\cdot)$ is the

Kullback–Leibler divergence between two distributions. We can optimize the objective function in Eqn. (2.6) w.r.t. the parameters ϕ and θ using back-propagation.

2.2 Experiments

We evaluate `DeepSTPP` for spatiotemporal prediction using both synthetic and real-world data.

2.2.1 Baselines

We compare `DeepSTPP` with the state-of-the-art models, including

- Spatiotemporal Hawkes Process (MLE) (Reinhart, 2018): it learns a spatiotemporal parametric intensity function using maximum likelihood estimation, see derivation in Appendix A.3.
- Recurrent Marked Temporal Point Process (RMTPP) (Du et al., 2016): it uses GRU to model the temporal intensity function. We modify this model to take spatial location as marks.
- Neural Spatiotemporal Point Process (NSTPP) (Chen et al., 2020): a neural point process model that parameterizes the spatial PDF and temporal intensity with continuous-time normalizing flows. Specifically, we use Jump CNF as it is a better fit for Hawkes processes.

All models are implemented in PyTorch, trained using the Adam optimizer. We set the number of representative points to be 100. The details of the implementation are deferred to the Appendix C.1. For the baselines, we use the authors' original repositories whenever possible.

2.2.2 Datasets

We simulated two types of STPPs: spatiotemporal Hawkes process (STH) and spatiotemporal self-correcting process (STSC) . For both STPPs, we generate three synthetic datasets, each with a different parameter setting, denoted as DS1, DS2, and DS3 in the tables. We also derive and implement efficient algorithms for simulating STPPs based on Ogata's thinning algorithm (Ogata, 1981). We view the simulator construction as an independent contribution from this work. The details of the simulation can be found in Appendix B. We use two real-world spatiotemporal event datasets from NSTPP (Chen et al., 2020) to benchmark the performance.

- **Earthquakes Japan**: catalog earthquakes data including the location and time of all earthquakes in Japan from 1990 to 2020 with magnitude of at least 2.5 from the U.S. Geological Survey. There are in total 1,050 sequences. The number of events per sequences ranges between 19 to 545 [1].
- **COVID-19**: daily county level COVID-19 cases data in New Jersey state published by The New York Times. There are 1,650 sequences and the number of events per sequences ranges between 7 to 305.

For both synthetic data and real-world data, we partition long event sequences into non-overlapping subsequences according to a fixed time range T. The targets are the last event, and the input is the rest of the events. The number of input events varies across subsequences. For each dataset, we split each into train/val/test sets with the ratio of 8:1:1. All results are the average of 3 runs.

[1]The statistics differ slightly from the original paper due to updates in the data source.

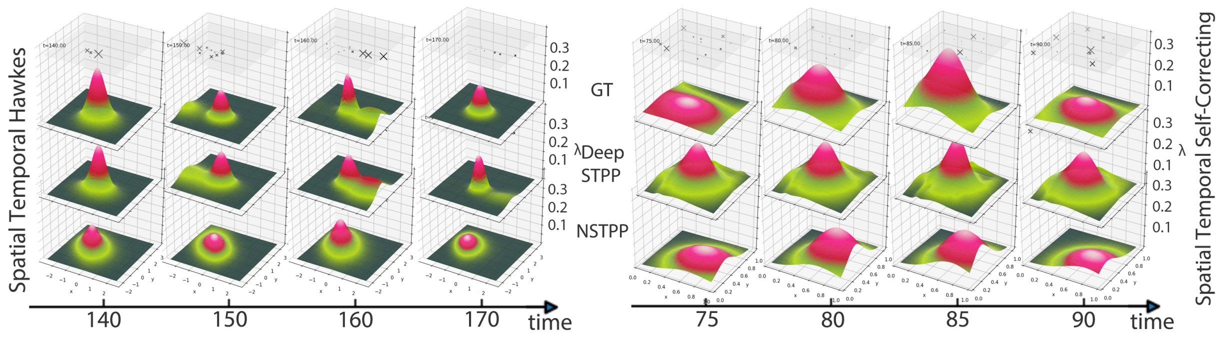


Figure 2.2: Ground-truth and learned intensity on two synthetic data. **Top**: ground-truth; **Middle**: learned intensity by our `DeepSTPP` model. **Bottom**: learned conditional intensity by NSTPP. 'X's refer to event history, where smaller 'X' refers to larger time difference.

2.2.3 Synthetic Experiment Results

For synthetic data, we know the ground truth intensity function. We compare our method with the best possible estimator: maximum likelihood estimator (MLE), as well as the NSTPP model. The MLE is learned by optimizing the log-likelihood using the BFGS algorithm. RMTPP can only learn the temporal intensity thus is not included in this comparison.

Predictive log-likelihood.

Table 2.1 shows the comparison of the predictive distribution for space and time. We report Log Likelihood (LL) of $f(\mathbf{s}, t|\mathcal{H}_t)$ and the Hellinger Distance (HD) between the predictive distributions and the ground truth averaged over time.

On both the STH and STSC datasets with different parameter settings, `DeepSTPP` outperform the baseline NSTPP in terms of LL and HD. It shows that `DeepSTPP` can estimate the spatiotemporal intensity more accurately for point processes with unknown parameters.

Table 2.1: Test log likelihood (LL) and Hellinger distance of distribution (HD) on synthetic data (LL higher is better, HD lower is better). Comparison between ours and NSTPP on synthetic datasets from two type of spatiotemporal point processes.

	Spatiotemporal Hawkes process					
	DS1		DS2		DS3	
	LL	HD	LL	HD	LL	HD
`DeepSTPP` (ours)	**-3.8420**	**0.0033**	**-3.1142**	**0.4920**	**-3.6327**	**0.0908**
NSTPP	-5.3110	0.5341	-4.8564	0.5849	-3.7366	0.1498
	Spatiotemporal Self Correcting process					
	DS1		DS2		DS3	
	LL	HD	LL	HD	LL	HD
`DeepSTPP` (ours)	**-1.2248**	**0.2348**	**-1.4915**	**0.1813**	**-1.3927**	**0.2075**
NSTPP	-2.0759	0.5426	-2.3612	0.3933	-3.0599	0.3097

Temporal intensity estimate.

Table 2.2 shows the mean absolute percentage error (MAPE) between the models' estimated temporal intensity and the ground truth $\lambda^*(t)$ over a short sampled range. On the STH datasets, since MLE has the correct parametric form, it is the theoretical optimum. Compared to baselines, `DeepSTPP` generally obtained the same or lower MAPE. It shows that joint spatiotemporal modeling also improve the performance of temporal prediction.

Intensity visualization.

Figure 2.2 visualizes the learned space-time intensity and the ground truth for STH and STSC, providing strong evidence that `DeepSTPP` can correctly learn the underlying dynamics of the spatiotemporal events. Especially, NSTPP has difficulty in modeling the complex dynamics of the multimodal distribution such as the spatiotemporal Hawkes process. NSTPP sometimes produces overly smooth intensity surfaces, and lost most of the details at the peak. In contrast, our `DeepSTPP` can better fit the multimodal distribution through the form of kernel summation and

Table 2.2: Estimated $\lambda^*(t)$ MAPE on synthetic data

	STH			STSC		
	DS1	DS2	DS3	DS1	DS2	DS3
DeepSTPP	3.33	369.44	11.30	**7.84**	**3.22**	20.98
NSTPP	53.41	17.69	3.85	99.99	39.33	37.39
RMTPP	263.83	729.78	**0.62**	45.55	21.26	37.46
MLE	**2.98**	**11.30**	4.38	27.38	18.20	**20.01**

obtain more accurate intensity functions.

Computational efficiency.

Figure 2.3 provides the run time comparison for the training between DeepSTPP and NSTPP for 100 epochs. To ensure a fair comparison, all experiments are conducted on 1 GTX 1080 Ti with Intel Core i7-4770 and 64 GB RAM. Our method is 100 times faster than NSTPP in training. It is mainly because our spatiotemporal kernel formulation has a close form of integration, which bypasses the complex and cumbersome numerical integration.

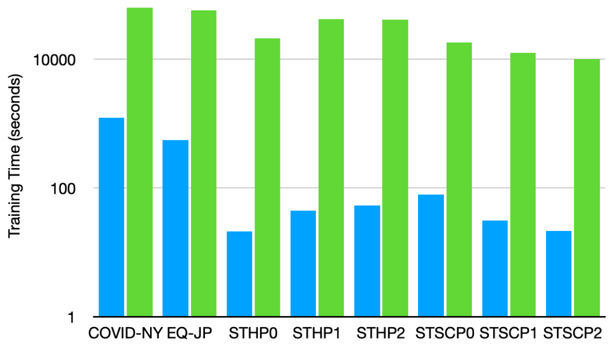


Figure 2.3: Log train time comparison on all datasets

Table 2.3: Test log likelihood (LL) comparison for space and time on real-world data over 3 runs.

LL	COVID-19 NY		Earthquake JP	
	Space	Time	Space	Time
DeepSTPP	$-0.1150_{\pm 0.0109}$	$2.4583_{\pm 0.0008}$	$\mathbf{-4.4025}_{\pm 0.0128}$	$\mathbf{0.4173}_{\pm 0.0014}$
NSTPP	$\mathbf{-0.0798}_{\pm 0.0433}$	$\mathbf{2.6364}_{\pm 0.0111}$	$-4.8141_{\pm 0.1165}$	$0.3192_{\pm 0.0124}$
RMTPP	-	$2.4476_{\pm 0.0039}$	-	$0.3716_{\pm 0.0077}$

2.2.4 Real-World Experiment Results

For real-world data evaluation, we report the conditional spatial and temporal log-likelihoods, i.e., $\log f^*(\mathbf{s}|t)$ and $\log f^*(t)$, of the final event given the input events, respectively. The total log-likelihood, $\log f^*(s,t)$, is the summation of the two values.

Predictive performances.

As our model is probabilistic, we compare against baselines models on the test predictive LL for space and time separately in Table 2.3. RMTPP can only produce temporal intensity thus we only include the time likelihood. We observe that DeepSTPP outperforms NSTPP most of the time in terms of accuracy. It takes only half of the time to train, as shown in Figure 2.3. Furthermore, we see that STPP models (first three rows) achieve higher LL compared with only modeling the time (RMTPP). It suggests the additional benefit of joint spatiotemporal modeling to increases the time prediction ability.

2.2.5 Ablation study

We conduct ablation studies on the model design. Our model assumes a global latent process z that governs the parameters $\{w_i, \beta_i, \gamma_i\}$ with separate decoders. We examine other alternative

Table 2.4: Test LL for alternative model designs over 3 runs

(higher the better)	COVID-19 NY		STH DS2	
	Space	Time	Space	Time
Shared decoders	$-0.1152_{\pm 0.0142}$	$2.4581_{\pm 0.0030}$	$-2.4397_{\pm 0.0170}$	$\mathbf{-0.6060}_{\pm 0.0381}$
Separate processes	**-0.1057**$_{\pm 0.0140}$	$2.4561_{\pm 0.0048}$	$-2.4291_{\pm 0.0123}$	$-0.7022_{\pm 0.0050}$
LSTM encoder	$-0.1162_{\pm 0.0102}$	$2.4554_{\pm 0.0035}$	$-2.4331_{\pm 0.0174}$	$-0.6845_{\pm 0.0252}$
`DeepSTPP`	$-0.1150_{\pm 0.0109}$	**2.4583**$_{\pm 0.0008}$	$\mathbf{-2.4289}_{\pm 0.0102}$	$-0.6853_{\pm 0.0145}$

designs experimentally. (1) *Shared decoders*: We use one shared decoder to generate model parameters. Shared decoders input the sampled z to one decoder and partition its output to generate model parameters.(2) *Separate process*: We assume that each of the $\{w_i, \beta_i, \gamma_i\}$ follows a separate latent process and we sample them separately. Separate processes use three sets of means and variances to sample $\{w_i, \beta_i, \gamma_i\}$ separately. (3) *LSTM encoder*: We replace the Transformer encoder with a LSTM module.

As shown in Table 2.4, we see that (1) *Shared decoders* decreases the number of parameters but reduces the performance. (2) *Separate process* largely increases the number of parameters but has negligible influences in test log-likelihood. (3) *LSTM encoder*: changing the encoder from Transformer to LSTM also results in slightly worse performance. Therefore, we validate the design of DeepNSTPP: we assume all distribution parameters are governed by one single hidden stochastic process with separate decoders and a Transformer as encoder.

Chapter 3

Automatic Integration for Point Process

A central concept in point processes is the *intensity function*, which indicates the expected rates of events occurrence. Specifically, given the event sequence $\mathcal{H}_t = \{(\mathbf{s}_1, t_1), \dots, (\mathbf{s}_n, t_n)\}_{t_n \leq t}$, the joint log-likelihood function of the observed events for point process is as follows:

$$\log p(\mathcal{H}_t) = \sum_{i=1}^{n} \log \lambda^*(t_i) - \int_0^t \lambda^*(\tau) d\tau \tag{3.1}$$

where $\lambda^\star$ is the optimal intensity function.

One fundamental difficulty of maximum likelihood estimation for point processes lies in the integral term of (3.1). As there is no closed-form solution for the integration computation, existing approaches often use approximation. For example, Neural Hawkes process (Mei and Eisner, 2016) relies on Monte Carlo sampling, which can have high variance. Furthermore, existing NPPs often report log-likelihood as a performance measure but fail to validate the learned intensity function. We found test log-likelihood may not be a good metric not only because it is not exact, but also

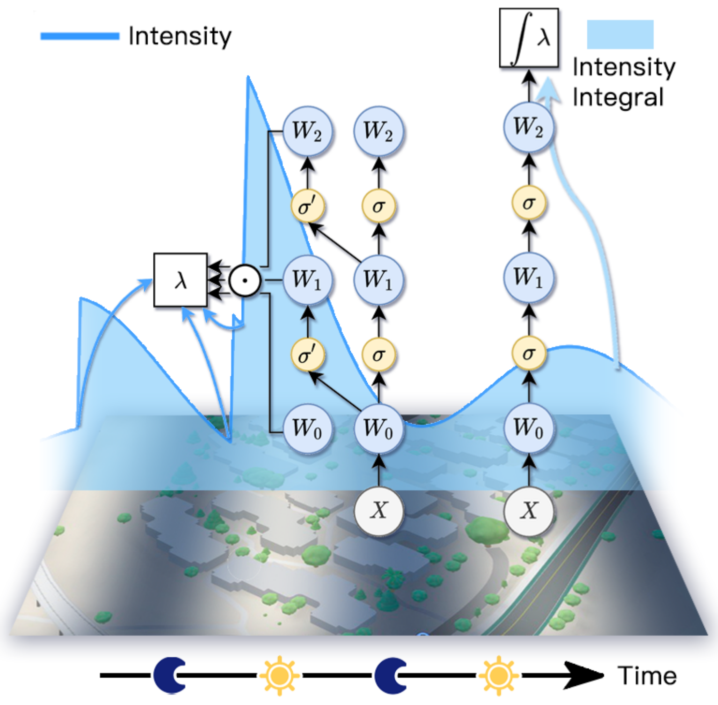


Figure 3.1: Illustration of learning point process with automatic integration. W denotes the linear layer's weight. σ is the nonlinear activation function. Left shows the intensity network that approximates λ and right is the integral network that computes $\int \lambda$. The two networks share the same weights.

because learning a wrong intensity function can have little to no effect on the test likelihood.

The expressivity of NPP is another limitation. Existing methods assume that the current influence follows an exponential decay of the intensity function (Du et al., 2016), or of the latent representation (Mozer et al., 2017a), or even a linear interpolation (Zuo et al., 2020). Such an assumption is easily violated in real-world scenarios. An example is the delayed effect in social media posts; the influence of a viral post will not appear until several hours later, which means there could be a jump in intensity that violates the smoothness assumption. In other cases, the event influence can be cyclic, e.g., a social media bot posts every day around the same time. Both scenarios turn out to be very challenging for the existing NPP models.

To reduce the cost of integration computation and improve the expressivity of the intensity function, we ask a natural question:

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

If successful, the resulting NPP would significantly relax the assumptions imposed by existing NPPs and open up new venues for modeling complex real-world event dynamics with “delayed jump” or “cyclic influence”. Unfortunately, such a strategy has a major bottleneck that has prevented others from pursuing further: it requires integrating a complicated deep neural network over a large time span, where numerical integration is both inefficient and erroneous.

In this chapter, we solve this problem based on the idea of automatic integration (Lindell et al., 2021; Li et al., 2019). We recognize that taking the partial derivative of a feed-forward network results in a new computational graph that shares the same set of parameters, see Figure 3.1. Regular NPP models use a neural network with a positive activation function to approximate the intensity. In contrast, we first construct a monotonically increasing integral network whose partial derivative is the intensity we wish to integrate. Then, we train the integral network to maximize the data likelihood. Finally, we reassemble the parameters of the integral network to obtain the intensity. This technique leads to exact solutions of the intensity and its antiderivative without imposing any constraints on their functional forms. As a result, we can efficiently compute the exact likelihood of *any* sophisticated intensity. We validate our approach using synthetic point process data with complex intensity functions, as well as a real-world earthquake dataset.

To summarize, our contributions are the following:

- We propose the first framework to speedup neural point processes learning with automatic integration. We use two networks and enforce the positivity of the intensity via a monotone integral network.

- We show that the automatic integration learns intensity functions with higher efficiency and accuracy than other NPP approaches
- We propose a simple NPP model that can recover complex influence functions, enjoys high training speed and better interpretability, and performs on par with the state-of-the-art methods on real-world data.

3.1 Methodology

In this section, we introduce a new design of the Neural Point Process, which is more interpretable and flexible. We explain how automatic integration can be used in conjunction with such an NPP for fast training and inference.

3.1.1 Limitations of Existing NPPs

Previous NPP models lack easy interpretation of event influence, and are not compatible with AutoInt. We begin by challenging the assumptions made by previous methods: Neural Hawkes process (Mei and Eisner, 2016) and RMTPP (Du et al., 2016).

Neural Hawkes process encodes the event history immediately after the n-th event as a hidden vector $\mathbf{h}(t_n^+) \in \mathbb{R}^k$. The model also predicts a scalar decay rate β_{t_n} and the hidden state immediately before the next event $\mathbf{h}(t_{n+1}^-)$. The model interpolates the two hidden state vectors using a kernel function f and maps the output to the intensity use a linear layer $\mathbf{w} \in \mathbb{R}^{1\times k}$. Thus, the conditional intensity function is formulated as

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

where g^+ is a positive activation function.

RMTPP also encodes the event history immediately after the n-th event as a hidden vector $\mathbf{h}(t_n^+)$, but it directly maps the vector to a scalar. The model also assumes the same intensity decay rate β applies to all events. It uses a scalar interpolation function f, and the resulting conditional intensity function is

$$\lambda^*(t)|_{t_n \le t \le t_{n+1}} = \mu + g^+(\mathbf{w}^T \mathbf{h}(t_n^+) + f(t - t_n, \beta)),$$

where μ is the scalar base intensity.

Neural Hawkes and RMTPP assume that each event's influence over the intensity is limited to the inter-event interval $[t_n^+, t_{n+1}^-]$. Their intensities are defined separately in each interval. Whereas for Hawkes process in (1.1), each event has a long-lasting influence; whenever a new event happens, we can decompose the summation in the intensity to analyze the contribution of any historical event to the happening of the new event, which is much more interpretable.

Both models also assume a simple change of intensity in each interval with an exponential function f. In the "cyclic influence" scenario as mentioned in Section , an event may have periodic reduction in influence. Either model performs poorly in this case. While the Neural Hawkes process can capture either increasing or decreasing influence, it cannot handle such a multi-modal intensity, see Figure 3.2. We verified that such failures are common in Section 3.2.

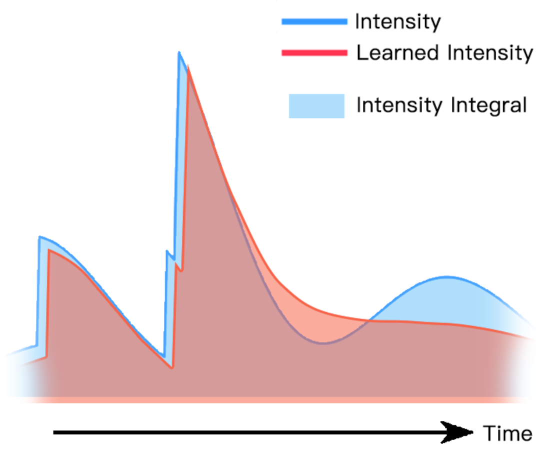


Figure 3.2: An example failure to restore the ground truth intensity. It is challenging because the ground truth and estimated likelihood are the same (the areas under the curves are the same.) The model needs not oversimplify the intensity to learn it correctly.

Furthermore, the designs of Neural Hawkes and RMTPP are not compatible with AutoInt. While AutoInt can approximate any function f and its antiderivative in closed forms, finding a closed-form antiderivative of $g^+ \circ f$ (where g^+ is a positive activation function) is still intractable.

3.1.2 Influence-Driven Point Process

We consider the following neural point process model generalizing the Hawkes process in (1.1):

$$\lambda^*(t) = \mu + \sum_{t_i < t} f_\theta^+(t - t_i, \mathcal{H}(t_i)). \tag{3.2}$$

where μ is the scalar base intensity. f_θ^+ is a positive scalar function that accepts time and $\mathcal{H}(t_i)$, i.e., a representation of the event history up to the i-th event, as inputs. The model is compatible with AutoInt as the intensity function does not contain function composition. As a result, it can efficiently evaluate definite integral over a long time span without constraining the influence to be a kernel function.

Our design has two major benefits. First, by constructing f_θ^+ as a deep neural network, our model can approximate *any* complex inter-event change of intensity, including the “cyclic influence" scenario as shown in Figure 3.2. Second, our model considers the long-lasting influence f of each individual event and represents the intensity in a more interpretable way. We can analyze and interpret different past events’ contribution percentages to a new event by decomposing the intensity function. Whereas the previous NPP models, the past influence is a black box as it is determined arbitrarily by the hidden representation.

There are many options for representing the input $\mathcal{H}(t_i)$ to the neural network. We can directly use the difference in event times as input:

$$\lambda^*(t) = \mu + \sum_i f_\theta^+(t - t_i), f_\theta : \mathbb{R}^1 \to \mathbb{R}^1 \tag{3.3}$$

Alternatively, we may use a sequence model such as an RNN to encode the history into hidden vectors $\{\mathbf{h}_i\}_{i=0}^N$. The hidden vectors could be then used to scale each event’s influence, such that the conditional intensity is formulated as

$$\lambda^*(t) = \mu + \sum_i g_\phi(\mathbf{h}_i) f_\theta^+(t - t_i), \tag{3.4}$$

$$f_\theta : \mathbb{R}^1 \to \mathbb{R}^1, \quad g_\phi : \mathbb{R}^k \to \mathbb{R}^1, \tag{3.5}$$

where g_ϕ is another neural network. We can also concatenate time t and $\mathbf{h}_i$ and feed them to the neural network, and the conditional intensity becomes

$$\lambda^*(t) = \mu + \sum_i f_\theta^+(t - t_i \oplus \mathbf{h}_i), \tag{3.6}$$

$$f_\theta : \mathbb{R}^{k+1} \to \mathbb{R}^1 \tag{3.7}$$

In practice, incorporating a deep sequence model greatly increases the flexibility of the model but can easily overfit. It also reduces the interpretability of the model as the influence function is in the high-dimensional domain.

3.1.3 Automatic Integration (AutoInt)

Suppose we have a scalar function $f_\theta(t, \mathbf{h})$ representing the intensity, we want to calculate $\int_{t=a}^{b} f_\theta(t, \mathbf{h}) := F_\theta(b, \mathbf{h}) - F_\theta(a, \mathbf{h})$ along one axis. AutoInt constructs the integral network F_θ first, and then reorganize the computational graph of F_θ to represent the integrant f_θ. The two networks thus share the same set of parameters θ.

Specifically, let $\mathbf{x} := t \oplus \mathbf{h}$, we consider the integral of the intensity as a fully-connected multi-layer neural network 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 of the neural network and σ denotes the elementwise nonlinearity. M_k and N_k are the input and output dimension for the k-th layer. Hence, the set of parameters in this neural network is $\theta = \{\mathbf{W}_k \in \mathbb{R}^{M_k \times N_k}, \forall k\}$.

The influence network f_θ is a partial derivative of the integral network F_θ. As long as the

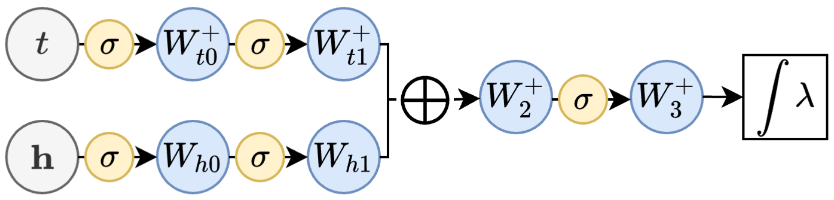


Figure 3.3: The architecture for monotonically increasing integral network that computes the the integral of the intensity. t is the time of the event and h is the encoded hidden vector. "W^+" indicates the neural network layer has non-negative weights.

activation function is differentiable everywhere, the intensity can be computed recursively:

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

$$\cdots \circ \mathbf{W}_2 \sigma'(\mathbf{W}_1\mathbf{x}) \circ \mathbf{W}_{11}$$

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

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

As noted, computing $f_\theta(\mathbf{x})$ involves many repeated operations. For example, the result of $\mathbf{W}_1\mathbf{x}$ is used for compute both $\sigma(\mathbf{W}_1\mathbf{x})$ and $\sigma'(\mathbf{W}_1\mathbf{x})$, see Figure 3.1. Therefore, we implemented a program that leverages dynamical programming to efficiently create a derivative model using automatic differentiation. During training, we use the two networks to calculate the likelihood of event sequences, as if they are regular neural networks.

3.1.4 Imposing the Non-negativity Constraint

We constrain the intensity to be non-negative by forcing its integral network to be monotonically increasing. As this is a strict constraint, we cannot use regularization by penalizing negative gradients; otherwise, the negative intensity would lead to erroneous log-likelihood. Also, we cannot simply constrain all linear weights in the network to be non-negative because we want the network to be monotonic only for the time input but not others.

We design the following architecture for the integral network, as illustrated in Figure 3.3: 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. Therefore, 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 also ensure the expressivity of other input dimensions is not impaired.

We experimented with different ways to enforce positive weights. We found that projected gradient descent (i.e., clamping the weights after each optimizer step) converges to the ground truth better than the exponential transformation method. To ensure monotonicity, we need to use monotonic activation function; previous AutoInt works use sine activation (Lindell et al., 2021) which is non-monotonic. We found that tanh and sine activations yield similar performance, as also indicated by Parascandolo et al. (2016).

3.1.5 Loss Function

Given the monotonic integral network $F_\theta(t, \mathbf{h})$ and the event influence network $f_\theta = \frac{\partial F_\theta}{\partial t}$ obtained from AutoInt, the log-likelihood of an event sequence $\{t_1, ..., t_N\}$ observed in time interval

$[0, T]$ with respect to the model is

$$\begin{aligned}
&\mathcal{L}(\{t_1, ..., t_N\}, \{\mathbf{h}_0, ..., \mathbf{h}_N\}) \\
=&\sum_{i=1}^{N} \log \left(\sum_{j=1}^{i-1} f_\theta(t_i - t_j, \mathbf{h}_i) \right) + \\
&\sum_{i=1}^{N} [F_\theta(T - t_i, \mathbf{h}_N) - F_\theta(0, \mathbf{h}_i)],
\end{aligned}$$

where $\{\mathbf{h}_0, ..., \mathbf{h}_N\}$ are the latent representations generated by a deep sequence model. This is straightforward by the Fundamental Theorem of Calculus. In case where the deep sequence model is not used, the log-likelihood evaluates to

$$\begin{aligned}
\mathcal{L}(\{t_1, ..., t_N\}) =&\sum_{i=1}^{N} \log \left(\sum_{j=1}^{i-1} f_\theta(t_i - t_j) \right) + \\
&\sum_{i=1}^{N} [F_\theta(T - t_i) - F_\theta(0)].
\end{aligned}$$

We can learn the parameters θ in both networks by maximizing the log-likelihood function.

3.2 Experiments

We evaluate `AIN` for learning temporal dynamics using both synthetic and real-world data.

3.2.1 Experimental Setup

Synthetic Datasets. Existing temporal point process models fail to capture the dynamics of the point process governed by multimodal or non-smooth current influence function. Here, we proposed and simulated three challenging synthetic datasets using Ogata's thinning algorithm (Chen,

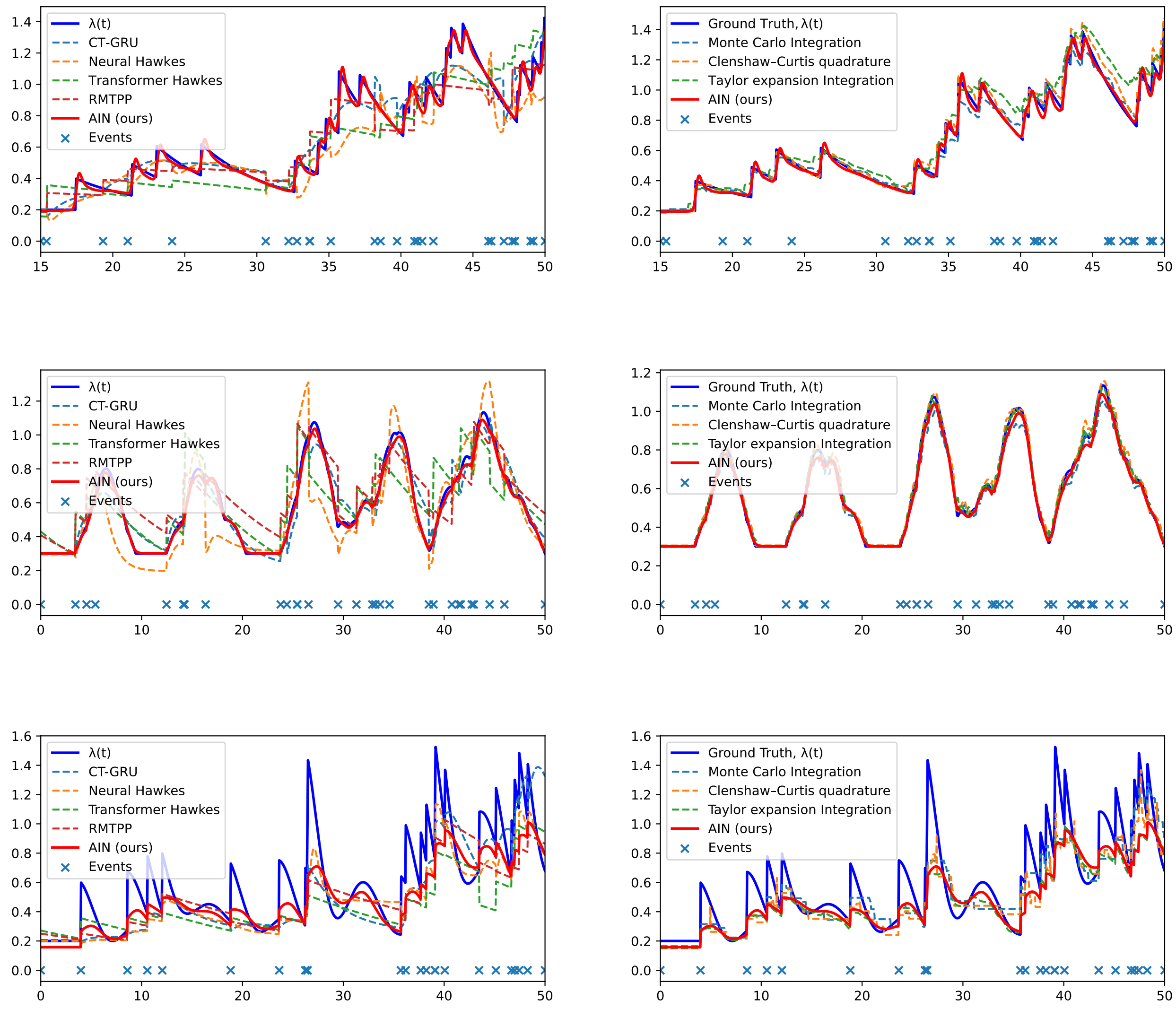


Figure 3.4: Visualizations of the true conditional intensity $\lambda^*(t)$ and the learned conditional intensity on the *Shift Hawkes* (first row), *Delayed Peak* (second row), *Shaky Hawkes* (third row) datasets. First column: comparison of intensities learned with different models. Second column: comparison of intensities learned with different integration methods.

Table 3.1: Comparison between our proposed model AIN (with or without RNN) and the state-of-the-art NPP models on three synthetic datasets and the *Earthquake Japan* dataset. Performance w.r.t. Mean Absolute Percentange Error (MAPE) of the estimated conditional intensity $\lambda^*(t)$ and Test log likelihood (LL).

Model	shakyHawkes		shiftHawkes		decayPeak		earthquakesJP
	MAPE	LL	MAPE	LL	MAPE	LL	LL
CT-GRU (Mozer et al., 2017a)	0.2243	-35.6063	0.1262	-39.7173	0.1103	-42.1959	5.9148
Neural Hawkes (Mei and Eisner, 2016)	0.2168	-35.4043	0.1473	-40.0411	0.1468	-42.5548	7.0620
RMTPP (Du et al., 2016)	0.2562	-35.6549	0.2630	-39.7893	0.2183	-42.7965	7.7663
Transformer Hawkes (Zuo et al., 2020)	0.2812	-36.1831	0.2316	-40.6717	0.2342	-43.3308	6.0588
AIN	**0.1843**	**-35.3762**	**0.0356**	**-39.3599**	**0.0226**	**-41.9678**	**8.6187**
AIN (w/ RNN)	0.3353	-37.9182	0.4675	-44.0076	0.1107	-42.3124	7.7730

2016).

- *Shaky Hawkes process*: as illustrated by Figure 3.2, it multiplies the influence function of the Hawkes process (see Equation 1.1) by a cyclic function, such that the intensity becomes multimodal in long inter-event intervals. It is characterized by the conditional intensity function

$$\lambda^*(t) = \mu + \alpha \sum_{i=1}^{N} \cos((t - t_i) + 1) \exp(-\beta(t - t_i))$$

  In our experiments, we set $\alpha = \beta = \mu = 0.2$.

- *Delayed Peak process*: it features a uni-modal but non-smooth influence function. Each event's influence is initially 0; then it first increases and then decreases, following a bell-shaped curve. It is characterized by

$$\lambda^*(t) = \mu + \alpha \sum_{i=1}^{N} \text{ReLU}(-(\beta(t - t_i) - 1)^2 + 1)$$

  In our experiments, we set $\alpha = 0.2, \beta = 0.5, \mu = 0.3$.

- *Shift Hawkes process*: as mentioned in Section , it describes the scenario in which a post becomes viral several hours after it is visible, such that there is a jump in the intensity between events. It is characterized by

$$\lambda^*(t) = \mu + \alpha \sum_{i=1}^{N} \mathbf{1}(t - t_i > \gamma) \exp(-\beta(t - t_i - \gamma))$$

In our experiments, we set $\alpha = \beta = \mu = 0.2$ and the threshold $\gamma = 2.0$.

Each synthetic dataset contains 8192 sequences over a time range of $[0, 50)$. The train-val-test split is $2 : 1 : 1$.

Real-world Datasets. We use a real-world dataset, *Earthquake Japan*, that includes the times and locations of all earthquakes in Japan from 1990 to 2020 with magnitudes of at least 2.5. It is gathered by Chen et al. (2020). The dataset contains 1500 sequences over a time range of $[0, 30)$. The train-val-test split is $4 : 1 : 1$.

Evaluation Metrics. As shown by Figure 3.2, a TPP model may yield a likelihood similar to the ground truth but fail to capture the correct intensity. Therefore, in addition to the likelihood (LL), we show the Mean Absolute Percentage Error (MAPE) of the estimated conditional intensity.

3.2.2 Baselines

We have two groups of baselines:

- *Integration methods*: the baselines learn the point process using the same model as described by Equation 3.3 but with four different integration techniques: Taylor integration ((Liu, 2020), see Appendix D.1), the Clenshaw–Curtis quadrature (see Appendix D.2), the Monte Carlo integration, and AutoInt.

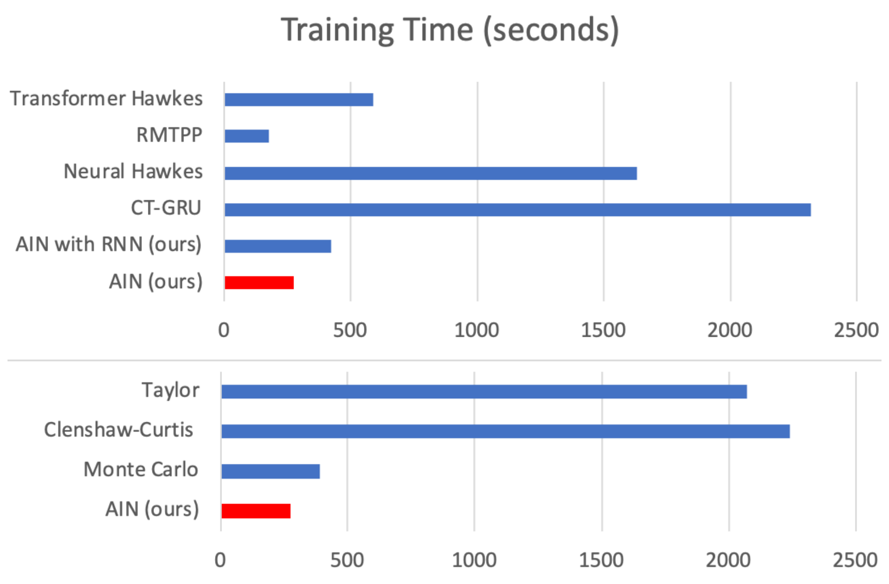


Figure 3.5: Training speed comparison for different NPPs and numerical integration methods in seconds. The proposed AIN is fast. RMTPP is the fastest but suffers from poor prediction performance.

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

3.2.3 Experimental Results

Figure 3.5 visualizes the run-time comparison of different methods. We can see that AutoInt trains faster than other numerical integration techniques, and AIN is much faster than most state-of-the-art models.

Table 3.1 compares the prediction Mean Absolute Percentage Error (MAPE) and test log-

Table 3.2: Comparison between different integration methods on three synthetic datasets and the *Earthquake Japan* dataset. Performance w.r.t. Mean Absolute Percentange Error (MAPE) of the estimated conditional intensity $\lambda^*(t)$ and Test log likelihood (LL).

Model	shakyHawkes		shiftHawkes		decayPeak		earthquakesJP
	MAPE	LL	MAPE	LL	MAPE	LL	LL
Clenshaw-Curtis	0.2197	-35.5183	0.0541	-39.4831	0.0312	-41.9839	8.4299
Monte Carlo	0.1935	-35.6090	0.0462	**-39.3527**	0.0378	-41.9868	8.1906
Taylor Expansion (Liu, 2020)	0.2004	-35.3771	0.0999	-39.7062	**0.0224**	-41.9691	7.7142
AIN	**0.1843**	**-35.3762**	**0.0356**	-39.3599	0.0226	**-41.9678**	**8.6187**

likelihood (LL) between AIN and the state-of-the-art models on the four datasets. Table 3.2 compares the evaluation metrics between AutoInt and other numerical integration methods using the same model. We can see that AIN has a decisive advantage on the synthetic dataset with complex intensity. We can also tell from the bottom-left of subfigure of Figure 3.4 that our method is the only one that can capture the multimodal intensity function.

Chapter 4

Related Works

4.1 Spatiotemporal Dynamics Learning.

Modeling the spatiotemporal dynamics of a system in order to forecast the future is a fundamental task in many fields. Most work on spatiotemporal dynamics has been focused on spatiotemporal data measured at regular space-time interval, e.g., (Xingjian et al., 2015; Li et al., 2018b; Yao et al., 2019; Fang et al., 2019; Geng et al., 2019). For discrete spatiotemporal events, statistical methods include space-time point process, see (Moller and Waagepetersen, 2003; Mohler et al., 2011). (Zhao et al., 2015) propose multi-task feature learning whereas (Yang et al., 2018) propose RNN-based model to predict spatiotemporal check-in events. These discrete-time models assume data are sampled evenly, thus are unsuitable for our task.

4.2 Continuous Time Sequence Models.

Continuous time sequence models provide an elegant approach for describing irregular sampled time series. For example, (Chen et al., 2018; Jia and Benson, 2019; Dupont et al., 2019; Gholami et al., 2019; Finlay et al., 2020; Kidger et al., 2020; Norcliffe et al., 2021) assumes the latent dynamics are continuous and can be modeled by an ODE. But for high-dimensional spatiotemporal processes, this approach can be computationally expensive. Che et al. (2018); Shukla and Marlin (2018) modifies the hidden states with exponential decay. GRU-ODE-Bayes proposed by De Brouwer et al. (2019) introduces a continuous-time version of GRU and a Bayesian update network capable of handling sporadic observations. However, Mozer et al. (2017b) shows that there is no significant benefit of using continuous-time RNN for discrete event data. Special treatment is still needed for modeling unevenly sampled events.

4.3 Deep Point Process.

Point process is well-studied in statistics (Moller and Waagepetersen, 2003; Daley and Vere-Jones, 2007; Reinhart, 2018). Deep point process couples deep learning with point process and has received considerable attention. For example, neural Hawkes process applies RNNs to approximate the temporal intensity function (Du et al., 2016; Mei and Eisner, 2017; Xiao et al., 2017; Zhang et al., 2020b), and (Zuo et al., 2020) employs Transformers. (Shang and Sun, 2019) integrates graph convolution structure. However, all existing works focus on temporal point processes without spatial modeling. For datasets with spatial information, they discretize the space and treat them as discrete “markers”. Okawa et al. (2019) extends Du et al. (2016) for spatiotemporal event prediction but they only predict the density instead of the next location and time of the event. Zhu et al. (2019) parameterizes the spatial kernel with a neural network embedding without consider the temporal sequence. Recently, Chen et al. (2020) propose neural spatiotemporal point process (NSTPP) which combines continuous-time neural networks with continuous-time normalizing flows to parameterize spatiotemporal point processes. However, this approach is quite computationally expensive, which requires evaluating the ODE solver for multiple time steps.

4.4 Nonparametric Inference for Point Process.

Fitting traditional TPP models such as Hawkes process to data points may have bad performance if the model is misspecified. To address this issue, non-parametric inference for TPP has been extensively studied in the statistical literature. 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 over time.

Recently, Neural Point Processes that combine TPP with neural networks has received considerable attention (Yan et al., 2018; Upadhyay et al., 2018; Huang et al., 2019; Omi and Aihara, 2019; Shang and Sun, 2019; Zhang et al., 2020a). The neural network enables the estimation of intensity after each event and significantly improves model flexibility. Under this framework, models focus more on approximating a discrete set of intensities before and after each event. The continuous intensity comes from interpolating the intensity points. 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. Other works consider alternative training schema: Xiao et al. (2017) used Wasserstein distance, Guo et al. (2018) introduced noise-contrastive estimation, and Li et al. (2018a) leveraged reinforcement learning. While these NPP models are more expressive than the traditional models, they still assume simple (continuous, usually monotonous) inter-event intensity changes.

4.5 Integration Methods.

Integration method is largely ignored in NPP literature, but is central to a model's ability to capture the complex dynamics of a system. Existing works either used an intensity function with an elementary integral (Du et al., 2016) or used Monte Carlo integration (Mei and Eisner, 2016). However, we can see from Figure 1.2 that the choice of integration method has a non-trivial effect on the model performance.

Integration is generally more complicated than differentiation, which can be mechanically solved using the chain rule. Most integration rules, e.g., integration by parts and change of variables, transform an antiderivative to another that is not necessarily easier. Elementary antiderivative only exists for a small set of functions, but not even for simple composite functions such as $\exp(x^2)$ (Dunham, 2018). The Risch algorithm can determine such elementary antiderivative (Risch, 1969, 1970) but has never been fully implemented due to its complexity. The most commonly used integration methods are still numerical: Newton-Cotes Methods, Romberg Integration, Quadrature, and Monte Carlo integration (Davis and Rabinowitz, 2007).

Multiple recent works claimed for launching a new integration approach, Automatic Integration (AutoInt). Liu (2020) proposes integrating the Taylor polynomial using the derivatives from Automatic Differentiation (AutoDiff). It requires partitioning of the integral limits and choosing the order of Taylor approximation. Though it makes use of 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 dual network approach which we will discuss in detail in Section 3. The method guarantees a closed-form integral and is efficient.

4.6 Monotonic Constraints.

We use AutoInt for NPP, but to enforce the intensity's nonnegativity, we have to constrain its integral to be monotonically increasing. Monotonicity in neural networks has been widely studied (Archer and Wang, 1993; Doumpos and Zopounidis, 2009; Sharma and Wehrheim, 2020). There are two groups of existing approaches: network architecture design and constraints in loss.

Network architecture design usually means constraining all signs of the linear layers' weights to be positive and having a monotonic activation function. Examples include applying an element-wise exponential transformation to all linear layers' weights (Sill, 1998) or learning weights using a projected stochastic gradient descent (Chorowski and Zurada, 2014). The non-monotonic heuristics are usually approximated by randomly sampling gradients from the input domain and penalizing negative samples using hinge loss (Liu et al., 2020).

Chapter 5

Conclusion

We propose two new paradigms for learning continuous-time dynamics behind irregularly sampled spatiotemporal events. Our first model, Deep Spatiotemporal Point Process (`DeepSTPP`), integrates a principled spatiotemporal point process with deep neural networks. We derive a tractable inference procedure by modeling the space-time intensity function as a composition of kernel functions and a latent stochastic process. We infer the latent process with neural networks following the variational inference procedure. Using synthetic data from the spatiotemporal Hawkes process and self-correcting process, we show that our model can learn the spatiotemporal intensity accurately and efficiently. We demonstrate superior forecasting performance on many real-world benchmark spatiotemporal event datasets. Our second model, Automatic Integration for Neural point process (`AIN`), uses a dual network approach. `AIN` can efficiently compute the exact likelihood of *any* sophisticated intensity. We validate our approach using many synthetic data with complex intensity functions, and a real-world earthquake dataset. Experiment results demonstrate that `AIN` can efficiently and accurately recover the underlying intensity function.

Challenges remain. While our models address improving the expressive power of kernel intensity estimation and learning a broader class of intensity function, they involve some simplified conditions. `DeepSTPP` assumes the influence is always centered at the event location. `AIN` cannot handle a process that has both self-correcting events (whose intensity is in product form) and self-exciting events, as integrating the product of neural networks is intractable. An interesting future direction is to extend this work to multivariate spatiotemporal processes, such that different dimensions assume different conditions. We hope our work's novelty will inspire new advancements in the field of the neural point process.

Appendix A

Model Details

A.1 Spatiotemporal Point Process Derivation

Conditional Density.

The intensity function and probability density function of STPP is related:

$$
\begin{aligned}
f(\mathbf{s}, t|\mathcal{H}_t) &= \frac{\lambda^*(\mathbf{s}, t)}{1 - F^*(\mathbf{s}, t)} \\
&= \lambda^*(\mathbf{s}, t) \exp\left(-\int_{\mathcal{S}} \int_{t_n}^{t} \lambda^*(\mathbf{s}, \tau) d\tau d\mathbf{s}\right) \\
&= \lambda^*(\mathbf{s}, t) \exp\left(-\int_{t_n}^{t} \lambda^*(\tau) d\tau\right)
\end{aligned}
$$

The last equation uses the relation that $\lambda^*(\mathrm{s}, t) = \lambda^*(t) f(\mathrm{s}|t)$, according Daley and Vere-Jones (2007) Chapter 2.3 (4). Here $\lambda^*(t)$ is the time intensity and $f^*(\mathrm{s}|t) := f(\mathrm{s}|t, \mathcal{H}_t)$ is the spatial PDF that the next event will be at location s given time t. According to Daley and Vere-Jones

(2007) Chapter 15.4, we can also view STPP as a type of TPP with continuous (spatial) marks,

Likelihood.

Given a STPP, the log-likelihood of observing a sequence $\mathcal{H}_t = \{(\mathbf{s}_1, t_1), (\mathbf{s}_2, t_2), ...(\mathbf{s}_n, t_n)\}_{t_n \le t}$ is given by:

$$\begin{aligned}
\mathcal{L}(\mathcal{H}_{t_n}) &= \log\left[\prod_{i=1}^{n} f(\mathbf{s}_i, t_i|\mathcal{H}_{t_{i-1}})(1 - F^*(\mathbf{s}, t))\right] \\
&= \sum_{i=1}^{n}\left[\log\lambda^*(\mathbf{s}_i, t_i) - \int_{\mathcal{S}}\int_{t_{i-1}}^{t_i}\lambda^*(\tau)d\tau d\mathbf{s}\right] + \log(1 - F^*(\mathbf{s}, t)) \\
&= \sum_{i=1}^{n}\log\lambda^*(\mathbf{s}_i, t_i) - \int_{\mathcal{S}}\int_{0}^{t_n}\lambda^*(\mathbf{s}, \tau)d\tau - \int_{\mathcal{S}}\int_{t_n}^{T}\lambda^*(\mathbf{s}, \tau)d\tau \\
&= \sum_{i=1}^{n}\log\lambda^*(\mathbf{s}_i, t_i) - \int_{\mathcal{S}}\int_{0}^{T}\lambda^*(\mathbf{s}, \tau)d\tau \\
&= \sum_{i=1}^{n}\log\lambda^*(t_i) + \sum_{i=1}^{n}\log f^*(\mathbf{s}_i|t_i) - \int_{0}^{T}\lambda^*(\tau)d\tau
\end{aligned}$$

Inference.

With a trained STPP and a sequence of history events, we can predict the next event timing and location using their expectations, which evaluate to

$$\begin{aligned}
\mathbb{E}[t_{n+1}|\mathcal{H}_{t_n}] &= \int_{t_n}^{\infty} t\int_{\mathcal{S}} f(\mathbf{s}, t|\mathcal{H}_{t_n})d\mathbf{s}dt = \int_{t_n}^{\infty} t\exp\left(-\int_{t_n}^{t}\lambda^*(\tau)d\tau\right)\int_{\mathcal{S}}\lambda^*(\mathbf{s}, t)d\mathbf{s}dt, \\
&= \int_{t_n}^{\infty} t\exp\left(-\int_{t_n}^{t}\lambda^*(\tau)d\tau\right)\lambda^*(t)dt \qquad \text{(A.1)}
\end{aligned}$$

The predicted location for the next event is:

$$\begin{aligned}\mathbb{E}[\mathbf{s}_{n+1}|\mathcal{H}_{t_n}] &= \int_{t_n}^{\infty} \mathbf{s} \int_{\mathcal{S}} \lambda^*(\mathbf{s}, t) \exp\left(-\int_{t_n}^{t} \lambda^*(\mathbf{s}, \tau) d\tau\right) d\mathbf{s} dt \\ &= \int_{t_n}^{\infty} \exp\left(-\int_{t_n}^{t} \lambda^*(\tau) d\tau\right) \int_{\mathcal{S}} \mathbf{s}\lambda^*(\mathbf{s}, t) d\mathbf{s} dt \end{aligned} \tag{A.2}$$

Computational Complexity.

It is worth noting that both learning and inference require conditional intensity. If the conditional intensity has no analytic formula, then we need to compute numerical integration over $\mathcal{S}$. Then, evaluating the likelihood or either expectation requires at least triple integral. Note that $\mathbb{E}[t_i|\mathcal{H}_{t_{i-1}}]$ and $\mathbb{E}[\mathbf{s}_i|\mathcal{H}_{t_{i-1}}]$ actually are sextuple integrals, but we can memorize all $\lambda^*(\mathrm{s}, t)$ from $t = t_{i-1}$ to $t \gg t_{i-1}$ to avoid re-compute the intensities. However, memorization leads to high space complexity. As a result, we generally want to avoid an intractable conditional intensity in the model.

A.2 Deep Spatiotemporal Point process (`DeepSTPP`) Derivation

PDF Derivation

The model design of `DeepSTPP` enjoys a closed form formula for the PDF. First recall that

$$f^*(t) = \lambda^*(t) \exp\left(-\int_{t_n}^{t} \lambda^*(\tau) d\tau\right)$$

Also notice that $f^*(\mathbf{s},t) = f^*(\mathbf{s}|t)f^*(t)$, $\lambda^*(\mathbf{s},t) = f^*(\mathbf{s}|t)\lambda^*(t)$ and $\lambda^*(t) = \dfrac{f^*(t)}{1-F^*(t)}$.

Therefore

$$\begin{aligned}
f^*(\mathbf{s},t) &= f^*(\mathbf{s} \mid t)f^*(t) \\
&= f^*(\mathbf{s} \mid t)\lambda^*(t)\exp\left(-\int_{t_n}^{t}\lambda^*(\tau)d\tau\right) \\
&= \lambda^*(\mathbf{s},t)\exp\left(-\int_{t_n}^{t}\lambda^*(\tau)d\tau\right)
\end{aligned}$$

For `DeepSTPP`, the spatiotemporal intensity is

$$\lambda^*(\mathbf{s},t) = \sum_i w_i \exp(-\beta_i(t-t_i))k_s\left(\mathbf{s}-\mathbf{s}_i\right)$$

The temporal intensity simply removes the k_s (which integrates to one). The bandwidth doesn't matter.

$$\lambda^*(t) = \sum_i w_i \exp(-\beta_i(t-t_i))$$

Integrate $\lambda^*(\tau)$ yields

$$\int \lambda^*(\tau)d\tau = -\sum_i \frac{w_i}{\beta_i}\exp(-\beta_i(\tau-t_i)) + C$$

Note that deriving the $\exp$ would multiply the coefficient $-\beta_i$. The definite integral is

$$\int_{t_n}^{t}\lambda^*(\tau)d\tau = -\sum_i \frac{w_i}{\beta_i}[\exp(-\beta_i(t-t_i)) - \exp(-\beta_i(t_n-t_i))]$$

Then replacing the integral in the original formula yields

$$
\begin{aligned}
f^*(\mathbf{s}, t) &= \lambda^*(\mathbf{s}, t) \exp\left(-\int_{t_n}^{t} \lambda^*(\tau) d\tau\right) \\
&= \lambda^*(\mathbf{s}, t) \exp\left(\sum_i \frac{w_i}{\beta_i} [\exp(-\beta_i(t - t_i)) - \exp(-\beta_i(t_n - t_i))]\right)
\end{aligned}
$$

The temporal kernel function $k_t(t, t_i) = \exp(-\beta_i(t - t_i))$, we reach the closed form formula.

Inference

The expectation of the next event time is

$$
\mathbb{E}^*[t_i] = \int_{t_{i-1}}^{\infty} t f^*(t) dt = \int_{t_n}^{\infty} t \lambda^*(t) \exp\left(-\int_{t_{i-1}}^{t} \lambda^*(\tau) d\tau\right) dt
$$

where the inner integral has a closed form. It requires 1D numerical integration.

Given the predicted time $\bar{t_i}$, the expectation of the space can be efficiently approximated by

$$
\mathbb{E}^*[\mathbf{s}_i] \approx \mathbb{E}^*[\mathbf{s}_i | \bar{t_i}] = \sum_{i' < i} \alpha^{-1} w_{i'} k_t(\bar{t_i}, t_{i'}) \mathbf{s}_{i'}
$$

where $\alpha = \sum_{i'<i} w_{i'} k_t(\bar{t_i}, t_{i'})$ is a normalize coefficient.

A.3 Spatiotemporal Hawkes Process Derivation

Spatiotemporal Hawkes process (STHP).

Spatiotemporal Hawkes (or self-exciting) process is one of the most well-known STPPs. It assumes every past event has an additive, positive, decaying, and spatially local influence over future events. Such a pattern resembles neuronal firing and earthquakes.

Spatiotemporal Hawkes is characterized by the following intensity function (Reinhart, 2018):

$$\lambda^*(\mathbf{s}, t) := \mu g_0(\mathbf{s}) + \sum_{i:t_i<t} g_1(t, t_i) g_2(\mathbf{s}, \mathbf{s}_i) : \mu > 0 \tag{A.3}$$

where $g_0(\mathbf{s})$ is the probability density of a distribution over $\mathcal{S}$, g_1 is the triggering kernel and is often implemented as the exponential decay function, $g_1(\Delta t) := \alpha \exp(-\beta \Delta t) : \alpha, \beta > 0$, and $g_2(\mathbf{s}, \mathbf{s}_i)$ is the density of an unimodal distribution over $\mathcal{S}$ centered at $\mathbf{s}_i$.

Maximum Likelihood.

For spatiotemporal Hawkes process, we pre-specified the model kernels $g_0(\mathbf{s})$ and $g_2(\mathbf{s}, \mathbf{s}_j)$ to be Gaussian:

$$g_0(\mathbf{s}) := \frac{1}{2\pi} |\Sigma_{g0}|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\mathbf{s} - \mathbf{s}_\mu) \Sigma_{g0}^{-1} (\mathbf{s} - \mathbf{s}_\mu)^T\right) \tag{A.4}$$

$$g_2(\mathbf{s}, \mathbf{s}_j) := \frac{1}{2\pi} |\Sigma_{g2}|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\mathbf{s} - \mathbf{s}_j) \Sigma_{g2}^{-1} (\mathbf{s} - \mathbf{s}_j)^T\right) \tag{A.5}$$

Specifically for the STHP, the second term in the STPP likelihood evaluates to

$$\begin{aligned}
\int_0^T \lambda^*(\tau)d\tau &= \mu T + \alpha \int_0^T \int_0^\tau e^{-\beta(\tau-u)} dN(u)d\tau \\
&(0 \leq u \leq \tau, 0 \leq \tau \leq T) \rightarrow (u \leq \tau \leq T, 0 \leq u \leq T) \\
&= \mu T + \alpha \int_0^T \int_u^T e^{-\beta(\tau-u)} d\tau dN(u) \\
&= \mu T - \frac{\alpha}{\beta} \int_0^T \left[e^{-\beta(T-u)} - 1\right] dN(u) \\
&= \mu T - \frac{\alpha}{\beta} \sum_{i=0}^{N} \left[e^{-\beta(T-t_i)} - 1\right]
\end{aligned}$$

Finally, the STHP log-likelihood is

$$\mathcal{L} = \sum_{i=1}^{n} \log \lambda^*(\mathbf{s}_i, t_i) - \mu T + \frac{\alpha}{\beta} \sum_{i=0}^{N} \left[e^{-\beta(T-t_i)} - 1\right]$$

This model has 11 scalar parameters: 2 for $\mathbf{s}_\mu$, 3 for Σ_{g0}, 3 for Σ_{g2}, α, β, and μ. We directly estimate $\mathbf{s}_\mu$ as the mean of $\{s_i\}_0^n$, and then estimate the other 9 parameters by minimizing the negative log-likelihood using the BFGS algorithm. T in the likelihood function is treated as t_n.

Inference

Based on the general formulas in Appendix A.1, and also note that for an STHP,

$$\begin{aligned}\int_{t_{i-1}}^{t} \lambda^*(\tau)d\tau &= \int_0^t \lambda^*(\tau)d\tau - \int_0^{t_{i-1}} \lambda^*(\tau)d\tau \\ &= \left\{\mu t - \frac{\alpha}{\beta}\sum_{j=0}^{i-1}\left[e^{-\beta(t-t_j)} - 1\right]\right\} - \left\{\mu t_{i-1} - \frac{\alpha}{\beta}\sum_{j=0}^{i-1}\left[e^{-\beta(t_{i-1}-t_j)} - 1\right]\right\} \\ &= \mu(t - t_{i-1}) - \frac{\alpha}{\beta}\sum_{j=0}^{i-1}\left[e^{-\beta(t-t_{i-1}+t_{i-1}-t_j)} - e^{-\beta(t_{i-1}-t_j)}\right] \\ &= \mu(t - t_{i-1}) - \frac{\alpha}{\beta}\left(e^{-\beta(t-t_{i-1})} - 1\right)\sum_{j=0}^{i-1}\left[e^{-\beta(t_{i-1}-t_j)}\right] \text{ and}\end{aligned}$$

$$\begin{aligned}\int_{\mathcal{S}} \mathbf{s}\mu g_2(\mathbf{s}, \mathbf{s}_\mu)d\mathbf{s} &= \mu\mathbf{s}_\mu \\ \int_{\mathcal{S}} \mathbf{s}\sum_{i=0}^{n} g_1(t, t_i)g_2(\mathbf{s}, \mathbf{s}_i)d\mathbf{s} = \sum_{i=0}^{n} g_1(t, t_i)\int_{\mathcal{S}} \mathbf{s}g_2(\mathbf{s}, \mathbf{s}_i)d\mathbf{s} &= \sum_{i=0}^{n} g_1(t, t_i)\mathbf{s}_i \\ \int_{\mathcal{S}} \mathbf{s}\lambda^*(\mathbf{s}, t)d\mathbf{s} &= \mu\mathbf{s}_\mu + \sum_{i=0}^{n} g_1(t, t_i)\mathbf{s}_i,\end{aligned}$$

we have

$$\begin{aligned}\mathbb{E}[t_i|\mathcal{H}_{t_{i-1}}] = \int_{t_{i-1}}^{\infty} t &\left(\mu + \alpha\sum_{j=0}^{i-1} e^{-\beta(t-t_j)}\right) \\ &\exp\left(\frac{\alpha}{\beta}\left(e^{-\beta(t-t_{i-1})} - 1\right)\sum_{j=0}^{i-1}\left[e^{-\beta(t_{i-1}-t_j)}\right] - \mu(t - t_{i-1})\right) dt \text{ and}\end{aligned}$$

$$\mathbb{E}[\mathbf{s}_i|\mathcal{H}_{t_{i-1}}] = \int_{t_{i-1}}^{\infty} \left(\mu \mathbf{s}_\mu + \alpha \sum_{j=0}^{i-1} e^{-\beta(t-t_j)} \mathbf{s}_j \right)$$
$$\exp\left(\frac{\alpha}{\beta} \left(e^{-\beta(t-t_{i-1})} - 1 \right) \sum_{j=0}^{i-1} \left[e^{-\beta(t_{i-1}-t_j)} \right] - \mu(t - t_{i-1}) \right) dt$$

Both require only 1D numerical integration.

Spatiotemporal Self-Correcting process (STSCP).

A lesser-known example is self-correcting spatiotemporal point process Isham and Westcott (1979). It assumes that the background intensity increases with a varying speed at different locations, and the arrival of each event reduces the intensity nearby. The next event is likely to be in a high-intensity region with no recent events.

Spatiotemporal self-correcting process is capable of modeling some regular event sequences, such as an alternating home-to-work travel sequence. It has the following intensity function:

$$\lambda^*(\mathbf{s}, t) = \mu \exp\left(g_0(\mathbf{s})\beta t - \sum_{i:t_i<t} \alpha g_2(\mathbf{s}, \mathbf{s}_i) \right) : \alpha, \beta, \mu > 0 \tag{A.6}$$

Here $g_0(\mathbf{s})$ is the density of a distribution over $\mathcal{S}$, and $g_2(\mathbf{s}, \mathbf{s}_i)$ is the density of an unimodal distribution over $\mathcal{S}$ centered at $\mathbf{s}_i$.

Appendix B

Simulation Details

In this appendix, we discuss a general algorithm for simulating any STPP, and a specialized algorithm for simulating an STHP. Both are based on an algorithm for simulating any TPP.

B.1 TPP Simulation

The most widely used technique to simulate a temporal point process is Ogata's modified thinning algorithm, as shown in Algorithm 1 Daley and Vere-Jones (2007) It is a rejection technique; it samples points from a stationary Poisson process whose intensity is always higher than the ground truth intensity, and then randomly discards some samples to get back to the ground truth intensity.

The algorithm requires picking the forms of $M^*(t)$ and $L^*(t)$ such that

$$\sup(\lambda^*(t + \Delta t), \Delta t \in [0, L(t)]) \leq M^*(t).$$

In other words, $M^*(t)$ is an upper bound of the actual intensity in $[t, t + L(t)]$. It is noteworthy that if $M^*(t)$ is chosen to be too high, most sampled points would be rejected and would lead to an inefficient simulation.

When simulating a process with decreasing inter-event intensity, such as the Hawkes process, $M^*(t)$ and $L^*(t)$ can be simply chosen to be $\lambda^*(t)$ and ∞. When simulating a process with increasing inter-event intensity, such as the self-correcting process, $L^*(t)$ is often empirically chosen to be $2/\lambda^*(t)$, since the next event is very likely to arrive before twice the mean interval length at the beginning of the interval. $M^*(t)$ is therefore $\lambda^*(t + L^*(t))$.

Algorithm 1 Ogata Modified Thinning Algorithm for Simulating a TPP

```
1:  **Input:** Interval $[0, T]$, model parameters
2:  $t \leftarrow 0, \mathcal{H} \leftarrow \emptyset$
3:  **while true do**
4:      Compute $m \leftarrow M(t|\mathcal{H})$ , $l \leftarrow L(t|\mathcal{H})$
5:      Draw $\Delta t \sim$ Exp$(m)$ (exponential distribution with mean $1/m$)
6:      **if** $t + \Delta t > T$ **then**
7:          **return** $\mathcal{H}$
8:      **end if**
9:      **if** $\Delta t > l$ **then**
10:         $t \leftarrow t + l$
11:     **else**
12:         $t \leftarrow t + \Delta t$
13:         Compute $\lambda = \lambda^*(t)$
14:         Draw $u \sim$ Unif$(0, 1)$
15:         **if** $\lambda/m > u$ **then**
16:             $\mathcal{H} = \mathcal{H} \cup t$
17:         **end if**
18:     **end if**
19: **end while**
```

B.2 STPP Simulation

It has been mentioned in Chapter 2 that an STPP can be seen as attaching the locations sampled from $f^*(\mathbf{s}|t)$ to the events generated by a TPP. Simulating an STPP is basically adding one step to Algorithm 1: sample a new location from $f^*(\mathbf{s}|t)$ after retaining a new event at t.

As for a spatiotemporal self-correcting process, neither $f^*(\mathbf{s},t)$ nor $\lambda^*(t)$ has a closed form, so the process's spatial domain has to be discretized for simulation. $\lambda^*(t)$ can be approximated by $\sum_{\mathbf{s}\in\mathcal{S}}\lambda^*(\mathbf{s},t)/|\mathcal{S}|$, where $\mathcal{S}$ is the set of discretized coordinates. $L^*(t)$ and $M^*(t)$ are chosen to be $2/\lambda^*(t)$ and $\lambda^*(t+L^*(t))$. Since $f^*(\mathbf{s}|t)$ is proportional to $\lambda^*(\mathbf{s},t)$, sampling a location from $f^*(\mathbf{s}|t)$ is implemented as sampling from a multinomial distribution whose probability mass function is the normalized $\lambda^*(\mathbf{s},t)$.

B.3 STHP Simulation

To simulate a spatiotemporal Hawkes process with Gaussian kernel, we mainly followed an efficient procedure proposed by Zhuang (2004), that makes use of the clustering structure of the Hawkes process and thus does not require repeated calculations of $\lambda^*(\mathbf{s},t)$.

Algorithm 2 Simulating spatiotemporal Hawkes process with Gaussian kernel

```
1: Generate the background events $G^{(0)}$ with the intensity $\lambda^*(\mathbf{s},t)=\mu g_0(\mathbf{s})$, i.e., simulate a homogenous Poisson process Pois$(\mu)$ and sample each event's location from a bivariate Gaussian distribution $\mathcal{N}(\mathbf{s}_\mu,\Sigma)$
2: $\ell=0, S=G^{(\ell)}$
3: while $G^\ell \neq \emptyset$ do
4:     for $i \in G^\ell$ do
5:         Simulate event $i$'s offsprings $O_i^{(\ell)}$ with the intensity $\lambda^*(\mathbf{s},t)=g_1(t,t_i)g_2(\mathbf{s},\mathbf{s}_i)$, i.e., simulate a non-homogenous stationary Poisson process Pois$(g_1(t,t_i))$ by Algorithm 1 and sample each event's location from a bivariate Gaussian distribution $\mathcal{N}(\mathbf{s}_i,\Sigma)$
6:         $G^{(\ell+1)}=\bigcup_i O_i^{(\ell)}, S=S\cup G^{(\ell+1)}, \ell=\ell+1$
7:     end for
8: end while
9: return $S$
```

B.4 Parameter Settings

For the synthetic dataset, we pre-specified both the STSCP's and the STHP's kernels $g_0(\mathbf{s})$ and $g_2(\mathbf{s},\mathbf{s}_j)$ to be Gaussian:

$$g_0(\mathbf{s}) := \frac{1}{2\pi}|\Sigma_{g0}|^{-\frac{1}{2}}\exp\left(-\frac{1}{2}(\mathbf{s}-[0,0])\Sigma_{g0}^{-1}(\mathbf{s}-[0,0])^T\right)$$
$$g_2(\mathbf{s},\mathbf{s}_j) := \frac{1}{2\pi}|\Sigma_{g2}|^{-\frac{1}{2}}\exp\left(-\frac{1}{2}(\mathbf{s}-\mathbf{s}_j)\Sigma_{g2}^{-1}(\mathbf{s}-\mathbf{s}_j)^T\right)$$

The STSCP is defined on $\mathcal{S}=[0,1]\times[0,1]$, while the STHP is defined on $\mathcal{S}=\mathbb{R}^2$. The STSCP's kernel functions are normalized according to their cumulative probability on $\mathcal{S}$. Table B.1 shows the simulation parameters. The STSCP's spatial domain is discretized as an 101×101 grid during the simulation.

Table B.1: Parameter settings for the synthetic dataset

		α	β	μ	Σ_{g0}	Σ_{g2}
ST-Hawkes	DS1	.5	1	.2	[.2 0; 0 .2]	[0.5 0; 0 0.5]
	DS2	.5	.6	.15	[5 0; 0 5]	[.1 0; 0 .1]
	DS3	.3	2	1	[1 0; 0 1]	[.1 0; 0 .1]
ST-Self Correcting	DS1	.2	.2	1	[1 0; 0 1]	[0.85 0; 0 0.85]
	DS2	.3	.2	1	[.4 0; 0 .4]	[.3 0; 0 .3]
	DS3	.4	.2	1	[.25 0; 0 .25]	[.2 0; 0 .2]

Appendix C

Experiment Details

In this section, we include experiment configurations and some additional experiment results.

C.1 Model Setup Details

For a better understanding of `DeepSTPP`, we list out the detailed hyperparameter settings in Table C.1. We use the same set of hyperparameters across all datasets.

Table C.1: Hyperparameter settings for training `DeepSTPP` on all datasets.

Name	Value	Description
Optimizer	Adam	Optimizer of the Transformer-VAE is set to Adam
Learning rate	-	0.01(Synthetic) / 0.015(Real World)
Momentum	0.9	-
Epoch	200	Train the VAE for 200 epochs for 1 step prediction
Batch size	128	-
Encoder: `nlayers`	3	Encoder is composed of a stack of 3 identical Transformer layers
Encoder: `nheads`	2	Number of attention heads in each Transformer layer
Encoder: d_{model}	128	3-tuple history is embedded to 128-dimension before fed into encoder
Encoder: d_{hidden}	128	Dimension of the feed-forward network model in each Transformer layer
Positional Encoding	Sinusoidal	Default encoding scheme in Vaswani et al. (2017)
Decoder: d_{hidden}	128	Decoders for $w_i, \beta_i,$ andγ_i are all MLPs with 2 hidden layers whose dim = 128
d_z	128	Dimension of the latent variable z as shown in Figure 2.1
J	50	Number of representative points as described in Section 2.1.3; 100 during
β	1e-3	Scale factor multiplied to the log-likelihood in VAE loss

Appendix D

Numerical Integration for Point Process

D.1 Taylor Integration

In this section, we briefly introduce the algorithm proposed by Liu (2020) and our implementation of it. Let $\beta_1, \ldots, \beta_5$ be real scalar hyperparameters with $\beta_1 \neq 0$. They can be viewed as the coefficients of the Taylor remainders. In practice, we set $\beta_1 = 1$ and $\beta_2 = \beta_3 = \beta_4 = \beta_5 = 0$. We tried different sets of β, but we found that their influence is trivial in most case. If the real numbers $A_1, ..., A_5$ are given by

$$\begin{cases} A_5 := \frac{(b-c)^6-(a-c)^6}{6\beta_1^5} \\ A_4 := \frac{(b-c)^5-(a-c)^5}{5\beta_1^4} - \frac{4\beta_2 A_5}{\beta_1}, \\ A_3 := \frac{(b-c)^4-(a-c)^4}{4\beta_1^3} - \frac{3\beta_2 A_4}{\beta_1} - 3\left(\frac{\beta_3}{\beta_1} + \frac{\beta_2^2}{\beta_1^2}\right) A_5, \\ A_2 := \frac{(b-c)^3-(a-c)^3}{3\beta_1^2} - \frac{2\beta_2 A_3}{\beta_1} - \left(\frac{2\beta_3}{\beta_1} + \frac{\beta_2^2}{\beta_1^2}\right) A_4 + \\ -2\left(\frac{\beta_4}{\beta_1} + \frac{\beta_2\beta_3}{\beta_1^2}\right) A_5, \\ A_1 := \frac{(b-c)^2-(a-c)^2}{2\beta_1} - \frac{\beta_2 A_2}{\beta_1} - \frac{\beta_3 A_3}{\beta_1} - \frac{\beta_4 A_4}{\beta_1} - \frac{\beta_5 A_5}{\beta_1}, \end{cases}$$

and $y_1(c), ..., y_5(c)$ are given by evaluation of function and its derivatives

$$\underbrace{f(c)}_{y_0} + \underbrace{\beta_1 f'(c)}_{y_1} \varepsilon + \underbrace{\left\{\beta_2 f'(c) + \frac{1}{2!}\beta_1^2 f^{(2)}(c)\right\}}_{y_2} \varepsilon^2 + \underbrace{\left\{\beta_3 f'(c) + \beta_1\beta_2 f^{(2)}(c)\right.}_{y_3} + \underbrace{\left.\frac{1}{3!}\beta_1^3 f^{(3)}(c)\right\}}_{y_3} \varepsilon^3$$
$$+ \underbrace{\left\{\beta_4 f'(c) + \left(\beta_1\beta_3 + \frac{1}{2}\beta_2^2\right) f^{(2)}(c) + \frac{1}{2}\beta_1^2\beta_2 f^{(3)}(c)\ \frac{1}{4!}\beta_1^4 f^{(4)}(c)\right\}}_{y_4} \varepsilon^4$$
$$+ \underbrace{\left\{\beta_5 f'(c) + (\beta_1\beta_4 + \beta_2\beta_3) f^{(2)}(c)\frac{1}{2}\left(\beta_1^2\beta_3 + \beta_1\beta_2^2\right) f^{(3)}(c) + \frac{1}{6}\beta_1^3\beta_2 f^{(4)}(c) + \frac{1}{5!}\beta_1^5 f^{(5)}(c)\right\}}_{y_5} \varepsilon^5$$

The function $\int_a^b f(x)$ can be approximated by $\sum_k \int_a^b y_k(x) A_k$. We calculated the n-th derivative using PyTorch AutoDiff. When using the Taylor Integration for learning point process, we equally split the time interval to subintervals of width 0.1 (which means at most 500 subintervals on the synthetic dataset) and evaluate the integral over each subinterval.

D.2 Clenshaw-Curtis Quadrature

In this section, we briefly introduce the Clenshaw-Curtis algorithm, which integrates a function $f(x)$ by first approximating it using a linear combination of Chebyshev polynomials $T_n(x)$, which follows the recurrence relationship

$$T_0(x) = 1, T_1(x) = x, T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x)$$

or generally,

$$T_n(x) = \cos\left(n \cos^{-1} x\right)$$

The roots of the Chebyshev polynomials are Chebyshev nodes, where

$$x_k = \cos\left(\frac{2k-1}{2n}\pi\right), \quad k = 1, \ldots, n$$

We can use affine transformation to map the integrant from $x \in [a, b]$ to $x \in [-1, 1]$ and evaluate $(b-a)\sum\int_{-1}^{1} T_n(x)$ as an estimator of $\int_a^b f(x)$. Consider the average of function evaluated at nodes, $y = f(\bar{x}_k)$. The Chebyshev coefficients are

$$c_0 = \frac{1}{K}\sum f(x_k), c_n|_{n\neq 0} = \frac{2}{K}\sum T_n(x_k)f(x_k),$$

where x_k are the Chebyshev nodes. The integral of a Chebyshev polynomial is

$$\begin{aligned}
\int T_n(x)dx &= \int T_n(\cos\theta)d\cos\theta \\
&= -\int \cos(n\theta)\sin\theta d\theta \\
&= -\frac{1}{2}\int (\sin((n+1)\theta) - \sin((n-1)\theta))d\theta \\
&= \frac{1}{2}\left(\frac{\cos((n+1)\theta)}{n+1} - \frac{\cos((n-1)\theta)}{n-1}\right) + \text{const.} \\
&= \frac{1}{2}\left(\frac{T_{n+1}(x)}{n+1} - \frac{T_{n-1}(x)}{n-1}\right) + \text{const.}
\end{aligned}$$

Finally, we sum of the integral of each Chebyshev polynomial to obtain an estimate of $\int_a^b f(x)$. In our implementation, we use a default number of 1000 Chebyshev nodes when integrating the influence function.

References

Adams, R. P., Murray, I., and MacKay, D. J. (2009). 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.

Archer, N. P. and Wang, S. (1993). Application of the back propagation neural network algorithm with monotonicity constraints for two-group classification problems. *Decision Sciences*, 24(1):60–75.

Che, Z., Purushotham, S., Cho, K., Sontag, D., and Liu, Y. (2018). Recurrent neural networks for multivariate time series with missing values. *Scientific reports*, 8(1):1–12.

Chen, R. T., Amos, B., and Nickel, M. (2020). Neural spatio-temporal point processes. *arXiv preprint arXiv:2011.04583*.

Chen, R. T., Rubanova, Y., Bettencourt, J., and Duvenaud, D. K. (2018). Neural ordinary differential equations. In *Advances in neural information processing systems*, pages 6571–6583.

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

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

Chung, J., Gülçehre, Ç., Cho, K., and Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555.

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

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

Davis, P. J. and Rabinowitz, P. (2007). *Methods of numerical integration*. Courier Corporation.

De Brouwer, E., Simm, J., Arany, A., and Moreau, Y. (2019). Gru-ode-bayes: Continuous modeling of sporadically-observed time series. In *Advances in Neural Information Processing Systems*, pages 7379–7390.

Doumpos, M. and Zopounidis, C. (2009). Monotonic support vector machines for credit risk rating. *New Mathematics and Natural Computation*, 5(03):557–570.

Du, N., Dai, H., Trivedi, R., Upadhyay, U., Gomez-Rodriguez, M., and Song, L. (2016). Recurrent marked temporal point processes: Embedding event history to vector. In *KDD*, pages 1555–1564.

Dunham, W. (2018). *The calculus gallery*. Princeton University Press.

Dupont, E., Doucet, A., and Teh, Y. W. (2019). Augmented neural odes. In *Advances in Neural Information Processing Systems*, pages 3140–3150.

Fang, S., Zhang, Q., Meng, G., Xiang, S., and Pan, C. (2019). Gstnet: Global spatial-temporal network for traffic flow prediction. In *IJCAI*, pages 2286–2293.

Finlay, C., Jacobsen, J.-H., Nurbekyan, L., and Oberman, A. M. (2020). How to train your neural ode. *arXiv preprint arXiv:2002.02798*.

Geng, X., Li, Y., Wang, L., Zhang, L., Yang, Q., Ye, J., and Liu, Y. (2019). Spatiotemporal multi-graph convolution network for ride-hailing demand forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3656–3663.

Gholami, A., Keutzer, K., and Biros, G. (2019). Anode: Unconditionally accurate memory-efficient gradients for neural odes. *arXiv preprint arXiv:1902.10298*.

Guo, R., Li, J., and Liu, H. (2018). Initiator: Noise-contrastive estimation for marked temporal point process. In *IJCAI*, pages 2191–2197.

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

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8):1735–1780.

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

Isham, V. and Westcott, M. (1979). A self-correcting point process. *Stochastic processes and their applications*, 8(3):335–347.

Jia, J. and Benson, A. R. (2019). Neural jump stochastic differential equations. In *NeurIPS*, pages 9847–9858.

Kidger, P., Morrill, J., Foster, J., and Lyons, T. (2020). Neural controlled differential equations for irregular time series. *NeurIPS*.

Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

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

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

Li, S., Xiao, S., Zhu, S., Du, N., Xie, Y., and Song, L. (2018a). Learning temporal point processes via reinforcement learning. *arXiv preprint arXiv:1811.05016*.

Li, Y., Yu, R., Shahabi, C., and Liu, Y. (2018b). Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In *ICLR*.

Liang, W., Zhang, W., and Wang, X. (2019). Deep sequential multi-task modeling for next check-in time and location prediction. In *International Conference on Database Systems for Advanced Applications*, pages 353–357. Springer.

Lindell, D. B., Martel, J. N., and Wetzstein, G. (2021). Autoint: Automatic integration for fast neural volume rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14556–14565.

Liu, K. (2020). Automatic integration. *arXiv e-prints*, pages arXiv–2006.

Liu, X., Han, X., Zhang, N., and Liu, Q. (2020). Certified monotonic neural networks. *arXiv preprint arXiv:2011.10219*.

Mei, H. and Eisner, J. (2016). The neural hawkes process: A neurally self-modulating multivariate point process. *arXiv preprint arXiv:1612.09328*.

Mei, H. and Eisner, J. (2017). The neural hawkes process: A neurally self-modulating multivariate point process. In *NeurIPS*.

Mohler, G. O., Short, M. B., Brantingham, P. J., Schoenberg, F. P., and Tita, G. E. (2011). Self-exciting point process modeling of crime. *Journal of the American Statistical Association*, 106(493):100–108.

Møller, J., Syversveen, A. R., and Waagepetersen, R. P. (1998). Log gaussian cox processes. *Scandinavian journal of statistics*, 25(3):451–482.

Moller, J. and Waagepetersen, R. P. (2003). *Statistical inference and simulation for spatial point processes*. CRC Press.

Mozer, M. C., Kazakov, D., and Lindsey, R. V. (2017a). Discrete event, continuous time rnns. *arXiv preprint arXiv:1710.04110*.

Mozer, M. C., Kazakov, D., and Lindsey, R. V. (2017b). Discrete event, continuous time rnns. *arXiv:1710.04110*.

Norcliffe, A., Bodnar, C., Day, B., Moss, J., and Lio, P. (2021). Neural ode processes. *ICLR*.

Ogata, Y. (1981). On lewis' simulation method for point processes. *IEEE transactions on information theory*, 27(1):23–31.

Okawa, M., Iwata, T., Kurashima, T., Tanaka, Y., Toda, H., and Ueda, N. (2019). Deep mixture point processes: Spatio-temporal event prediction with rich contextual information. In *KDD*, pages 373–383.

Omi, T. and Aihara, K. (2019). Fully neural network based model for general temporal point processes. In *Advances in Neural Information Processing Systems*, pages 2122–2132.

Parascandolo, G., Huttunen, H., and Virtanen, T. (2016). Taming the waves: sine as activation function in deep neural networks.

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

Rehfeld, K., Marwan, N., Heitzig, J., and Kurths, J. (2011). Comparison of correlation analysis techniques for irregularly sampled time series. *Nonlinear Processes in Geophysics*, 18(3):389–404.

Reinhart, A. (2018). A review of self-exciting spatio-temporal point processes and their applications. *Statistical Science*, 33(3):299–318.

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

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

Safikhani, A., Kamga, C., Mudigonda, S., Faghih, S. S., and Moghimi, B. (2018). Spatio-temporal modeling of yellow taxi demands in new york city using generalized star models. *International Journal of Forecasting*.

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

Sharma, A. and Wehrheim, H. (2020). Testing monotonicity of machine learning models. *arXiv* *preprint arXiv:2002.12278*.

Shchur, O., Türkmen, A. C., Januschowski, T., and Günnemann, S. (2021). Neural temporal point processes: A review. *arXiv preprint arXiv:2104.03528*.

Shukla, S. N. and Marlin, B. (2018). Interpolation-prediction networks for irregularly sampled time series. In *ICLR*.

Sill, J. (1998). Monotonic networks.

Upadhyay, U., De, A., and Gomez-Rodriguez, M. (2018). Deep reinforcement learning of marked temporal point processes. *arXiv preprint arXiv:1805.09360*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Veen, A. and Schoenberg, F. P. (2008). Estimation of space–time branching process models in seismology using an em–type algorithm. *Journal of the American Statistical Association*, 103(482):614–624.

Xiao, S., Farajtabar, M., Ye, X., Yan, J., Song, L., and Zha, H. (2017). Wasserstein learning of deep generative point process models. In *NeurIPS*, pages 3247–3257.

Xingjian, S., Chen, Z., Wang, H., Yeung, D.-Y., Wong, W.-K., and Woo, W.-c. (2015). Convolutional lstm network: A machine learning approach for precipitation nowcasting. In *NeurIPS*, pages 802–810.

Yan, J., Liu, X., Shi, L., Li, C., and Zha, H. (2018). Improving maximum likelihood estimation of temporal point process via discriminative and adversarial learning. In *IJCAI*, pages 2948–2954.

Yang, G., Cai, Y., and Reddy, C. K. (2018). Recurrent spatio-temporal point process for check-in time prediction. In *CIKM*, pages 2203–2211. ACM.

Yao, H., Tang, X., Wei, H., Zheng, G., and Li, Z. (2019). Revisiting spatial-temporal similarity: A deep learning framework for traffic prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 5668–5675.

Zhang, Q., Lipani, A., Kirnap, O., and Yilmaz, E. (2020a). Self-attentive hawkes process. In *International Conference on Machine Learning*, pages 11183–11193. PMLR.

Zhang, Q., Lipani, A., Kirnap, O., and Yilmaz, E. (2020b). Self-attentive hawkes processes. In *International Conference on Machine Learning (ICML)*.

Zhao, L., Sun, Q., Ye, J., Chen, F., Lu, C.-T., and Ramakrishnan, N. (2015). Multi-task learning for spatio-temporal event forecasting. In *KDD*, pages 1503–1512.

Zhu, S., Li, S., Peng, Z., and Xie, Y. (2019). Imitation learning of neural spatio-temporal point processes. *arXiv preprint arXiv:1906.05467*.

Zuo, S., Jiang, H., Li, Z., Zhao, T., and Zha, H. (2020). Transformer hawkes process. *International Conference on Machine Learning (ICML)*.