Master's Thesis

Decoupled Marked Temporal Point Process using Neural ODEs

Yujee Song (송 유지)

Graduate School of Artificial Intelligence

Pohang University of Science and Technology

2025

신경 상미분방정식을 사용한 분리된 표시 시간점 프로세스

Decoupled Marked Temporal Point Process using Neural ODEs

Decoupled Marked Temporal Point Process using Neural ODEs

by

Yujee Song
Graduate School of Artificial Intelligence
Pohang University of Science and Technology

A thesis submitted to the faculty of the Pohang University of Science and Technology in partial fulfillment of the requirements for the degree of in the Artificial Intelligence

Pohang, Korea
11. 27. 2024
Approved by
Won Hwa Kim (Signature)
Academic advisor

Decoupled Marked Temporal Point Process using Neural ODEs

Yujee Song

The undersigned have examined this thesis and hereby certify that it is worthy of acceptance for a master's degree from POSTECH

11. 27. 2024

Committee Chair	Won Hwa Kim	(Seal)
Member	Dongwoo Kim	(Seal)
Member	Sungsoo Ahn	(Seal)

MGSAI 송 유지 Yujee Song
20232277
Decoupled Marked Temporal Point Process using Neural ODEs.
신경 상미분방정식을 사용한 분리된 표시 시간점 프로세스.
Graduate School of Artificial Intelligence , 2025, 37p
Advisor : Won Hwa Kim.
Text in English.

ABSTRACT

A Marked Temporal Point Process (MTPP) is a type of stochastic process, where its realization is a set of event-time data. MTPP is often used to analyze complex dynamics of discrete temporal events such as money transaction, social media, healthcare, etc. Recent studies have incorporated various deep neural networks to capture complex dependencies between events and generate embeddings that effectively represent the observed events. While most of existing research focuses on modeling inter-event dependencies and their representations, the influence of individual events on the overall temporal dynamics has received less attention. To address this gap, we introduce a Decoupled MTPP framework that separates the characterization of the stochastic process into a series of evolving influences from individual events. Our method uses Neural Ordinary Differential Equations (Neural ODEs) to learn the continuous dynamics of these influences flexibly, while simultaneously solving multiple

inference tasks. We demonstrate the importance of disentangling these influences by comparing our framework with state-of-the-art techniques on real-world datasets. We also provide analysis of the model's behavior, highlighting its potential applications.

Contents

I. Introduction **1**

II. Background **4**

2.1 Marked Temporal Point Process 4

2.1.1 Temporal Point Process (TPP) 4

2.1.2 Marked Temporal Point Process (MTPP) 4

2.1.3 Hawkes Process 5

2.2 Neural Ordinary Differential Equations 5

III. Method **7**

3.1 Decoupling Marked Temporal Point Processes 7

3.1.1 Hidden State Dynamics 8

3.1.2 Ground Intensity Function 9

3.1.3 Integration and Prediction via Neural ODEs 9

3.1.4 Conditional Probability for Marks 10

3.2 Linear Dec-ODE 10

3.2.1 Combining Influences from Past Events 10

3.2.2 Training Objective 11

3.2.3 Training Scheme 11

IV. Related Works **13**

4.1 Intensity Modeling 13

4.2 Direct Estimation of the PDF 13

4.3 Learning Temporal Dynamics with Neural ODEs 14

V. Experiment **15**

5.1 Comparison on Benchmarks 15

5.1.1 Experiment Setup 15

5.1.2 Comparison of Results 16

5.1.3 Explainability of Dec-ODE 17

5.1.4 Parallel Computing Scheme 19

5.2 Linear vs. Non-Linear Influence Function 19

5.3 Imputation 21

5.4 Simulation Study 22

5.5 Train time comparison 23

5.6 Implementation details 24

5.6.1 IVP solving with varying time intervals 24

5.6.2 Batch Computation 24

5.6.3 Baseline 25

5.6.4 Thinning algorithm 25

5.6.5 Training Details 26

5.7 Dataset Description 28

5.7.1 Benchmark dataset 28

5.7.2 Data preprocess 28

5.8 Visualizations 29

VI. Conclusion **31**

Summary (in Korean) **32**

References **34**

List of Tables

5.1 Comparison with the state of the art methods on 5 popular real-life datasets. Results with **boldface** and underline represent the best and the second-best results, respectively. Following [1, 2] the mean and std are gained by bootstrapping 1000 times. 16

5.2 Training efficiency comparison in different datasets with the average time taken for an iteration (sec/iter) as the metric. Ratio shows that the iteration time at least reduces in half. 19

5.3 Experiment on non-linear Φ_λ. The softplus is applied after the summation in order to express inhibitory effects. 20

5.4 Estimation accuracy on the simulated Hawkes process. 23

5.5 Training time (sec / epoch) compared between THP, ANHP, and Dec-ODE. 23

5.6 Required memory (MB) compared between baseline methods with batch size 4. 24

5.7 Statistics of benchmark datasets used for comparisons. 28

List of Figures

3.1 Overview of the proposed framework. Hidden states corresponding to each event e_i are individually propagated and decoded into trajectories $\mu(t; e_i)$ and $\hat{f}_k(k|t, e_i)$. These trajectories represent the influence of individual events on the MTPP, which is reconstructed by aggregating all the trajectories. 7

3.2 Visualization of (3.5). Different combinations of $\mu(t; e_i)$ can be selected for calculating $\lambda_g^*(t)$ and $f^*(t)$ conditioned on different $\mathcal{H}_{t_i}$ in parallel. 10

5.1 Visualization of propagated $\hat{f}^*(k|t, e_i)$ in the StackOverflow experiment. The axes represent time, event type, and magnitude, respectively. 17

5.2 Visualization of patterns in the Retweet dataset. (a) $\mu(t; e_i)$ for different marks, (b) Interaction between marks, (c) Event marks over time. 18

5.3 Imputation experiment done using Stackoverflow dataset, where from 10% to 40% of data are randomly dropped. 21

5.4 Visualization from a simulation study. (a), (b), (c), and (d) compare the true intensity function of a Hawkes process and the reconstructed results from THP, ANHP, and Dec-ODEs. Each $\mu(t)$ that composes the result in (d) is illustrated in (e). 22

5.5 Visualization of dynamics of μ and $\hat{f}$ in benchmark dataset. (a) is a visualization of ground intensity trained on MOOC dataset, (b) is a visualization of $\hat{f}(t; e_i)$ changing through time trained using MOOC dataset, (c) is a visualization of ground intensity trained on Reddit dataset, and (d) is a visualization of $\hat{f}(t; e_i)$ changing through time trained using Reddit dataset. 29

5.5 Visualization of dynamics of μ and $\hat{f}$ in benchmark dataset. (a) shows dynamics of ground intensity trained using Stack Overflow dataset, and (b) shows $\hat{f}(t; e_i)$ changing through time trained using MIMIC-II dataset. 30

I. Introduction

Event-time data, ubiquitous across domains like social media [3, 4], healthcare [5, 6], and stock market trends [7, 8], consist of sequences of events occurring at specific times and often categorized by type. To model such temporal data as a stochastic process, the goal is to predict the time and type of future events based on prior observations, i.e., the history of the sequence. For instance, analyzing a person's purchase history may help forecast when they will make their next purchase, while a brief post from an influential figure on social media could trigger significant movements in stock prices. This prediction often translates to estimating the likelihood of an event e occurring at time t, represented as a probability density function (pdf) $f(e, t)$.

Temporal Point Processes (TPP) have traditionally been used for such event predictions, where a realization of the stochastic process is a sequence of ordered time points [9]. By treating the event times as random variables, TPPs model the occurrence times of events based on past timestamps. The Hawkes Process, a well-known TPP model, captures these dynamics by employing a *conditional intensity function* that aggregates excitation effects from past events, a characteristic known as "self-excitation" [10, 11]. This framework independently models the influence of each event and combines them using a simple summation, making it both interpretable and computationally efficient.

Although many TPP-based models effectively handle the dynamic prediction of future events, they often overlook details such as spatial information, node-specific attributes (particularly in graph-based contexts like social or sensor networks), and additional context such as text, tokens, or images. Marked Temporal Point Processes (MTPP) address this limitation by introducing "marks," which represent event types, extending the traditional univariate TPP to account for multivariate event types. Recent MTPP approaches frequently combine the intensity functions of Hawkes Processes

with neural networks to flexibly model the pdfs of marks [12, 13, 14, 15, 2]. Prominent examples include Recurrent Marked Temporal Point Process [5], Neural Hawkes Process [12], and Transformer Hawkes Process [1], which effectively capture MTPP dynamics.

While these models excel at capturing complex dependencies, they often sacrifice interpretability of the relationships between events. Many existing approaches either directly estimate the joint probability distribution over time and marks, or use a proxy intensity function that does not fully capture the dynamic interaction between different marks and timestamps. Additionally, reconstructing the mark pdf or other related functions like the survival function typically involves complex integrations without closed-form solutions, requiring re-computation for each new event during inference. This results in computational inefficiencies, often scaling exponentially with the number of events.

In this context, we introduce a novel framework for MTPP modeling called Dec-ODE, characterized by decoupled hidden state dynamics and driven by latent continuous dynamics via neural ordinary differential equations (ODEs).

Dec-ODE decouples the hidden state dynamics for each event, allowing the model to independently quantify each event's contribution to the overall stochastic process with high fidelity. Moreover, this decoupling leverages ODEs to enable parallel training of continuous hidden state dynamics, significantly reducing computational overhead compared to sequential approaches that rely on step-by-step event updates.

Our Dec-ODE formulation offers the following contributions:

- Dec-ODE decouples individual events within a stochastic process for efficient learning and explainability of each mark,
- Dec-ODE models the continuous dynamics of influences from individually decoupled events by leveraging Neural ODEs,
- Dec-ODE demonstrates state-of-the-art performance on various benchmarks for

MTPP validation while effectively capturing inter-time dynamics with interpretability.

II. Background

2.1 Marked Temporal Point Process

2.1.1 Temporal Point Process (TPP)

A Temporal Point Process is a stochastic process where realizations are represented as ordered timestamps $\{t_i\}_{i=0}^{N}$. The sequence of events observed up to a specific time t_i is referred to as the history $\mathcal{H}_{t_i}$. The probability density function (pdf) $f(t|\mathcal{H}_{t_i}) := f^*(t)$ defines the likelihood of the next event occurring within the interval $[t, t + dt)$, conditioned on the history $\mathcal{H}_{t_i}$. Correspondingly, the cumulative density function (cdf) is $F(t|\mathcal{H}_{t_i})$, and the survival function is given by $S(t|\mathcal{H}_{t_i}) = 1 - F(t|\mathcal{H}_{t_i})$, both conditioned on the history $\mathcal{H}_{t_i}$. Throughout this paper, the superscript $*$ is used to indicate conditioning on $\mathcal{H}_{t_i}$.

Defining an explicit form for $f^*(t)$ is often challenging due to its unknown structure and the normalization constraint $\int_0^\infty f^*(s)\, ds = 1$. As a result, the conditional intensity function $\lambda^*(t) := \frac{f^*(t)}{S^*(t)} = \frac{f^*(t)}{1-F^*(t)}$ is commonly introduced to describe the stochastic process [9]. Since $\lambda^*(t) > 0$ is the only restriction, it offers flexibility in modeling TPPs [10, 12, 16, 13, 14, 1, 15, 2]. The conditional intensity function $\lambda^*(t)$ uses $\mathcal{H}_t$ to estimate the likelihood of an event occurring at time t, where $\lambda^*(t)dt = p(t_i \in [t, t + dt)|\mathcal{H}_t)$.

2.1.2 Marked Temporal Point Process (MTPP)

A Marked Temporal Point Process is a stochastic process that produces a sequence of events $\mathcal{S} = \{e_i\}_{i=0}^{N}$, where each event $e_i = (t_i, k_i)$ consists of a timestamp $t_i \in \mathbb{R}$ and a mark $k_i \in \{1, \ldots, K\}$, representing its type. The timestamps satisfy $t_i < t_{i+1}$. The prior events up to time t_i form the history $\mathcal{H}_{t_i} = \{(t_j, k_j) \in \mathcal{S} :$

$t_j < t_i\}$. For MTPPs, the conditional intensity function $\lambda^*(t,k) := \lambda(t,k|\mathcal{H}_t)$ can be expressed as a product of the ground intensity $\lambda_g^*(t)$ and the conditional probability of a mark $f^*(k|t)$ given time, as follows [9, 17, 18]:

$$\begin{aligned}\lambda^*(t,k) &= \lambda_g^*(t) f^*(k|t) \\ &= \frac{f(t|\mathcal{H}_t) f^*(k|t)}{1 - F(t|\mathcal{H}_t)} = \frac{f(t,k|\mathcal{H}_t)}{1 - F(t|\mathcal{H}_t)}.\end{aligned} \tag{2.1}$$

Here, $f^*(t)$ represents the temporal probability, while $f^*(k|t)$ specifies the conditional probability of a mark. Since $\lambda^*(t,k)$ is separable into $\lambda_g^*(t)$ and $f^*(k|t)$, the likelihood of an MTPP can be expressed as follows [18]:

$$f(t,k) = \left[\prod_{i=1}^{\mathcal{N}_g(t_N)} \lambda_g^*(t_i)\right] \left[\prod_{i=1}^{\mathcal{N}_g(t_N)} f^*(k_i|t_i)\right] \exp\left(-\int_0^{t_N} \lambda_g^*(u)\, du\right). \tag{2.2}$$

Here, $\mathcal{N}_g$ represents the counting process described by the ground intensity function $\lambda_g^*(t)$, where event marks are disregarded.

2.1.3 Hawkes Process

The Hawkes Process is a popular choice for modeling point processes exhibiting self-excitatory behavior, where past events increase the likelihood of future occurrences. Formally, its conditional intensity function $\lambda^*(t)$ is defined as [11]:

$$\lambda^*(t) = v + \sum_{t_i < t} \mu(t - t_i), \tag{2.3}$$

where t represents the current time, v is a non-negative baseline intensity, and $\mu(t)$ is an excitation function capturing the influence of past events at t_i on future event rates. The Hawkes Process has been widely studied in machine learning [2, 12, 1, 14] and applied in diverse fields [8, 19, 20].

2.2 Neural Ordinary Differential Equations

Neural Ordinary Differential Equations (Neural ODEs) combine the flexibility of neural networks with the mathematical rigor of differential equations [21], enabling

continuous-depth models in place of traditional fixed-layer architectures. These models define hidden state transformations as continuous flows governed by ODEs:

$$\frac{d\mathbf{z}(t)}{dt} = \gamma(\mathbf{z}(t), t|\theta), \tag{2.4}$$

where $\gamma(\cdot|\theta)$ is a neural network parameterized by θ, approximating the time evolution of the hidden state $\mathbf{z}(t)$. In our framework, Neural ODEs are used to model the continuous dynamics of how each event influences the overall MTPP.

III. Method

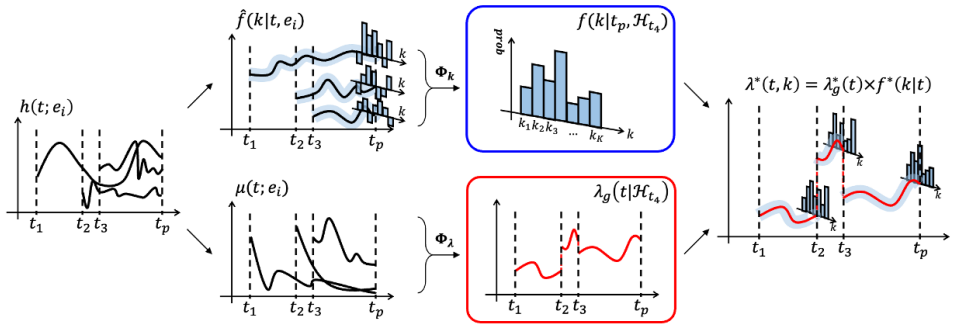


Figure 3.1: Overview of the proposed framework. Hidden states corresponding to each event e_i are individually propagated and decoded into trajectories $\mu(t;e_i)$ and $\hat{f}_k(k|t,e_i)$. These trajectories represent the influence of individual events on the MTPP, which is reconstructed by aggregating all the trajectories.

3.1 Decoupling Marked Temporal Point Processes

This work introduces a generalizable framework designed to effectively model MTPP distributions through decoupled structures. For a sequence of events $\mathcal{S} = \{e_i\}_{i=0}^n$, where each event $e_i = (t_i, k_i)$ consists of a timestamp t_i and a marker k_i, the objective is to predict the distribution of the next event e_{n+1}. Rather than directly estimating the conditional intensity $\lambda^*(t,k)$ as in prior approaches, we adopt a decoupled modeling strategy by separately estimating the ground intensity $\lambda_g^*(t)$ and the event distribution $f^*(k|t)$.

In our framework, the temporal influence of each event is captured by hidden state dynamics $h(t;e_i)$, which are decoded to yield $\lambda_g^*(t)$ and $f^*(k|t)$. As shown in Fig. 3.1, these two components are modeled as separate trajectories and are combined

to recover $\lambda^*(t, k)$.

3.1.1 Hidden State Dynamics

The framework incorporates hidden state dynamics that are decoupled for each event. These hidden states characterize the impact of an event e_i on the overall process. The decoupled hidden states, $h(t; e_i) \in \mathbb{R}^d$ for $i = 1, \cdots, n$, evolve independently over time from the occurrence of each event at t_i. These dynamics are parameterized using Neural ODEs [21] to model the intricate behavior of the intensity function in MTPP.

The initial state $h(t_i; e_i)$ depends on the event type k_i, and the dynamics are computed by solving an initial value problem (IVP). The hidden state for event e_i at time t is given as:

$$dh(t; e_i) = \gamma(h(t; e_i), t, k_i; \theta)dt, \tag{3.1}$$

$$\begin{aligned} h(t; e_i) &= h(t_i; e_i) + \int_{t_i}^{t} \gamma(h(s; e_i), s, k_i; \theta)ds \\ &= W_e(k_i) + \int_{t_i}^{t} \gamma(h(s; e_i), s, k_i; \theta)ds, \end{aligned} \tag{3.2}$$

where $W_e(\cdot) : \{1, \cdots, K\} \to \mathbb{R}^d$ is a trainable embedding, and γ is a neural network parameterized by θ. Since $h(t; e_i)$ is decoupled, hidden states at time t can be computed in parallel as a multidimensional ODE:

$$\frac{d}{dt}\mathbf{h}(t) = \frac{d}{dt}\begin{bmatrix} h(t; e_0) \\ \vdots \\ h(t; e_i) \end{bmatrix} = \begin{bmatrix} \gamma(h(t; e_0), t, k_0; \theta) \\ \vdots \\ \gamma(h(t; e_i), t, k_i; \theta) \end{bmatrix}. \tag{3.3}$$

The decoupled structure enables selective use of hidden states when computing $f^*(t)$ under specific histories $\mathcal{H}_t$, such as omitting $h(t; e_{j+1})$ when $t > t_{j+1}$. Detailed usage is described in Sec. 3.1.2.

3.1.2 Ground Intensity Function

The ground intensity $\lambda_g^*(t)$ is modeled as a combination of event-specific *influence functions* $\mu(t; e_i)$, which generalize the excitation functions in Hawkes processes [11]. To compute $\lambda_g(t|\mathcal{H}_t)$, the influences of historical events are aggregated:

$$\lambda_g(t|\mathcal{H}_{t_{n+1}}) = \Phi_\lambda(\mu(t; e_0), \cdots, \mu(t; e_n)), \tag{3.4}$$

where $\mu(t; e_i) := g_\mu(h(t; e_i))$, decoded via a neural network $g_\mu : \mathbb{R}^d \to \mathbb{R}$. The aggregation function Φ_λ ensures non-negativity. Decoding $\mu(t; e_i)$ prior to aggregation ensures visibility of individual event influences.

3.1.3 Integration and Prediction via Neural ODEs

The computation of various quantities such as likelihood, survival rates, and expected values in MTPP necessitates integration. Neural ODEs, being computationally demanding due to sequential solvers, are leveraged efficiently by solving multi-dimensional differential equations:

$$\frac{\partial}{\partial t}\begin{bmatrix} \mathbf{h}(t) \\ \Lambda_g(t|\mathcal{H}_{t_i}) \\ F(t|\mathcal{H}_{t_i}) \\ \mathbb{E}[t] \end{bmatrix} = \begin{bmatrix} \gamma(\mathbf{h}(t), t, \mathbf{k}; \theta) \\ \lambda_g(t|\mathcal{H}_{t_i}) \\ f(t|\mathcal{H}_{t_i}) \\ t \cdot f(t|\mathcal{H}_{t_i}) \end{bmatrix} = \begin{bmatrix} \gamma(\mathbf{h}(t), t, \mathbf{k}; \theta) \\ \Phi_\lambda(g_\mu(\mathbf{h}(t))) \\ \lambda_g(t|\mathcal{H}_{t_i}) \cdot \exp(\Lambda_g(t_{i-1}|\mathcal{H}_{t_i}) - \Lambda_g(t|\mathcal{H}_{t_i})) \\ t \cdot f(t|\mathcal{H}_{t_i}) \end{bmatrix} \tag{3.5}$$

This eliminates the need for separate integration steps, enabling simultaneous computation of multiple quantities, as visualized in Fig. 3.2. Therefore, important estimations with integration, such as likelihood, survivor rate $S^*(t) := 1 - F^*(t)$, and $\mathbb{E}_{f^*}[t]$ can be predicted in a single run. This formulation can be applied to any number of integrations.

Moreover, as briefly mentioned in Sec. 3.1.1, the influence functions can be selected according to our interests as illustrated in Fig. 3.2. Therefore, approximations of functions in (3.5) under different $\mathcal{H}_{t_i}$ can be obtained in parallel, distinguishing Dec-ODE from other intensity-based methods that require separate runs.

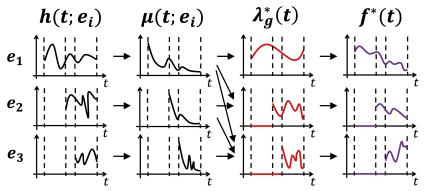


Figure 3.2: Visualization of (3.5). Different combinations of $\mu(t;e_i)$ can be selected for calculating $\lambda_g^*(t)$ and $f^*(t)$ conditioned on different $\mathcal{H}_{t_i}$ in parallel.

3.1.4 Conditional Probability for Marks

The conditional probability of marks $f^*(k|t)$ is formulated as:

$$f(k|t,\mathcal{H}_{t_{n+1}}) = \Phi_k(\hat{f}(k|t,e_0),\cdots,\hat{f}(k|t,e_n)), \tag{3.6}$$

where $\hat{f}(k|t,e_i) := g_f(h(t;e_i))$ and Φ_k ensures $\sum_K \Phi_k(\cdot) = 1$. This approach captures temporal dynamics in the event context, such as changing probabilities over time.

3.2 Linear Dec-ODE

3.2.1 Combining Influences from Past Events

To demonstrate the proposed framework's effectiveness, we implement a linear variant of Dec-ODE. Here, Φ_λ and Φ_k are defined as simple linear combinations while satisfying the constraints using activation functions:

$$\lambda_g(t|\mathcal{H}_{t_{n+1}}) = \sum_{e_i\in\mathcal{H}_{t_{n+1}}} \text{softplus}(\mu(t;e_i)), \tag{3.7}$$

$$f(k|t,\mathcal{H}_{t_{n+1}}) = \text{softmax}\left(\sum_{e_i\in\mathcal{H}_{t_{n+1}}} \hat{f}(k|t,e_i)\right). \tag{3.8}$$

The linearity offers interpretability and scalability while supporting complex temporal dynamics beyond classical Hawkes processes.

3.2.2 Training Objective

The goal of our approach is to maximize the likelihood of the predicted Marked TPP [18]. Taking the logarithm of the likelihood in (2.2), we obtain:

$$\begin{aligned}\ln f(t,k) &= \ln\left[\prod_{i=1}^{\mathcal{N}_g(t_N)} \lambda_g^*(t_i)\right]\left[\prod_{i=1}^{\mathcal{N}_g(t_N)} f^*(k_i|t_i)\right]\exp\left(-\int_0^{t_N}\lambda_g^*(u)du\right) \\ &= \underbrace{\sum_{i=1}^{\mathcal{N}_g(t_N)}\ln\lambda_g^*(t_i) - \int_0^{t_N}\lambda_g^*(u)du}_{\ln L_\lambda} + \underbrace{\sum_{i=1}^{\mathcal{N}_g(t_N)}\ln f^*(k_i|t_i)}_{\ln L_k},\end{aligned} \tag{3.9}$$

where t_N is the final observed time point. The log-likelihood terms $\ln L_\lambda$ for the ground intensity and $\ln L_k$ for the mark distribution are defined separately and are used to learn $\lambda_g(t)$ and $f(k|t)$, respectively.

Conceptually, the first term in L_λ captures the likelihood of events occurring at each t_i, while the second term accounts for the likelihood of events not occurring at all other times. To maximize L_λ, $\lambda_g^*(t)$ should be high at event occurrence times and low otherwise. Additionally, for estimation purposes, $f(t)$ is normalized to satisfy $\int f(t)dt = 1$.

3.2.3 Training Scheme

Prior works leveraging differential equation-based modeling [13, 15] typically solve over the entire time range sequentially, resulting in substantial training time. In contrast, our approach mitigates this limitation by utilizing the properties of Φ_λ with linear summation, as described below.

First, since each $\mu(t;e_i)$ evolves independently of other events, they can be propagated simultaneously by solving a multi-dimensional differential equation:

$$d\begin{bmatrix}\mu(\tau_0;e_0)\\ \vdots \\ \mu(\tau_i;e_i)\end{bmatrix} = \begin{bmatrix}\gamma(\mu(\tau_0),\tau_0,k_0;\theta)\\ \vdots \\ \gamma(\mu(\tau_i),\tau_i,k_i;\theta)\end{bmatrix}\cdot d\boldsymbol{\tau}, \tag{3.10}$$

where $\boldsymbol{\tau} = [\tau_0, \cdots, \tau_i]^\top = [t_0 + t, t_1 + t, \cdots, t_i + t]^\top$.

Second, formulating the ground intensity as a linear equation offers the advantage of computing the compensator $\Lambda_g^*(t)$ in a similar manner to (3.7):

$$\Lambda_g^*(t) = \int_0^t \lambda_g^*(s)ds = \int_0^t \sum_{e_i \in \mathcal{H}_t} \text{softplus}(\mu(s; e_i))ds \tag{3.11}$$

$$= \sum_{e_i \in \mathcal{H}_t} \int_{t_i}^t \text{softplus}(\mu(s; e_i))ds, \tag{3.12}$$

where the integration bounds shift because the influence of e_i begins only after t_i.

Equation (3.12) illustrates that the integration in (3.9) can be performed by independently integrating $\text{softplus}(\mu(t; e_i))$ for each event and subsequently summing the results, rather than sequentially processing the entire sequence. Consequently, given a fixed number of time points m, the full trajectory can be solved in a fixed number of steps. The efficiency of this approach is further discussed in Sec. 5.1.4.

IV. Related Works

4.1 Intensity Modeling

The intensity function is widely used in TPP modeling due to its flexibility in capturing temporal distributions, with deep learning enabling the parameterization of complex processes through neural networks. Early research often focused on relaxing the constraints of traditional TPPs such as the Hawkes process [10, 22, 18]. RNN-based neural TPP models [12, 5] leveraged RNNs to effectively encode event sequences into a history vector. Transformer-based approaches [1, 14] accounted for both long-term and short-term event dependencies in $\mathcal{H}_t$. The Attentive Neural Hawkes Process (ANHP) [2] further incorporated self-attention to model complex continuous temporal dynamics by applying attention mechanisms on unobserved events. In another direction, [23] utilized Gaussian processes to relax the exponential decay assumption in the Hawkes process. Additionally, [16] proposed using a compensator function—an integral of the intensity function—to alleviate computational challenges associated with integration in the absence of a closed form.

4.2 Direct Estimation of the PDF

Another approach involves directly predicting the probability density function (PDF) of a TPP. For instance, [24] proposed modeling the temporal distribution using a log-normal mixture. The closed-form PDF provided by this method enables direct inference of key characteristics, while the mixture model offers flexible modeling capabilities. Other approaches to modeling the PDF directly have incorporated popular deep learning frameworks such as GANs, normalizing flows, variational autoencoders, and meta-learning [25, 26].

4.3 Learning Temporal Dynamics with Neural ODEs

Neural ODEs, introduced in [21], treat changes in the hidden vector between layers as continuous, offering a novel framework for modeling temporal dynamics. Differential equation-based methods have since been applied to a range of temporal dynamic modeling tasks [27, 28, 29], demonstrating advantages in handling sparse and irregular time series. For instance, [13] used Neural ODEs to model jump stochastic differential equations and tested this on TPP modeling. [15] extended this work to spatio-temporal point processes, predicting both temporal and spatial dynamics using Neural ODEs and continuous normalizing flows, and proposed a batch-wise computation algorithm for time-varying sequences.

V. Experiment

5.1 Comparison on Benchmarks

This section presents a comparison between linear Dec-ODE and state-of-the-art methods using standard benchmarks.

5.1.1 Experiment Setup

Datasets. We evaluated our approach on five widely-used real-world datasets from various domains: **Reddit** represents sequences of social media posts, **Stack Overflow (SO)** comprises sequences of rewards from the question-answering platform, **MIMIC-II** contains sequences of diagnoses from clinical visits to the Intensive Care Unit (ICU), **MOOC** includes sequences of user interactions with an online course, and **Retweet** captures sequences of social media posts. Detailed descriptions of these datasets are provided in the Sec. 5.7. To ensure comparability across results, all time points were normalized by the standard deviation of the time gaps, $\{t_i - t_{i-1}\}_{i=1}^n$, within the training set, following the procedure outlined in [26].

Metrics. We employed three conventional metrics for evaluation: 1) *Negative Log-Likelihood* (NLL) assesses the suitability of the predicted probability density, as computed via (3.9); 2) *Root Mean Squared Error* (RMSE) evaluates the accuracy of predicted event time points; 3) *Accuracy* (ACC) quantifies the correctness of the predicted mark distribution $f^*(k|t)$. For predictions, the expected value $\mathbb{E}_{f^*}[t]$ was used as the predicted time of the next event, and the mark with the highest probability at time t, i.e., $\arg\max(f^*(k|t))$, was selected as the predicted mark.

Baseline. We compared our method against four state-of-the-art approaches: Recurrent Marked Temporal Point Process (RMTPP) [5], Transformer Hawkes Process (THP) [1], Intensity-Free Learning (IFL) [24], and Attentive Neural Hawkes Pro-

Table 5.1: Comparison with the state of the art methods on 5 popular real-life datasets. Results with **boldface** and underline represent the best and the second-best results, respectively. Following [1, 2] the mean and std are gained by bootstrapping 1000 times.

	RMSE					ACC					NLL				
	RMTPP	THP	IFL	ANHP	**Dec-ODE**	RMTPP	THP	IFL	ANHP	**Dec-ODE**	RMTPP	THP	IFL	ANHP	**Dec-ODE**
MOOC	0.473	0.476	0.501	0.470	**0.467**	20.98	24.49	32.30	31.53	**42.08**	−0.315	0.733	**-2.895**	−2.632	−2.289
	(0.012)	(0.010)	(0.012)	(0.019)	**(0.012)**	(0.29)	(0.22)	(1.30)	(0.20)	**(0.44)**	(0.031)	(0.047)	**(0.031)**	(0.043)	(0.191)
Reddit	0.953	6.151	1.040	1.149	**0.934**	29.67	60.72	48.91	**63.45**	62.32	3.559	2.335	2.188	**1.203**	1.367
	(0.016)	(0.195)	(0.017)	(0.010)	**(0.017)**	(1.19)	(0.08)	(1.27)	**(0.16)**	(0.11)	(0.070)	(0.031)	(0.088)	**(0.068)**	(0.126)
Retweet	0.990	1.055	1.012	1.663	**0.985**	51.72	**60.68**	55.35	59.72	60.17	−2.180	−2.597	−2.672	**-3.134**	−2.897
	(0.016)	(0.015)	(0.018)	(0.014)	**(0.016)**	(0.33)	**(0.11)**	(0.19)	(0.11)	(0.23)	(0.025)	(0.016)	(0.023)	**(0.019)**	(0.030)
MIMIC-II	0.859	> 10	1.005	0.933	**0.810**	78.20	**85.98**	80.49	84.30	85.06	1.167	5.657	**0.939**	1.025	1.354
	(0.093)	(0.114)	(0.121)	(0.088)	**(0.173)**	(5.00)	**(2.56)**	(5.20)	(2.78)	(3.65)	(0.150)	(0.304)	**(0.139)**	(0.155)	(0.413)
Stack	**1.017**	1.057	1.340	1.052	1.018	53.95	53.83	53.00	**56.80**	55.58	2.156	2.318	2.314	**1.873**	2.063
Overflow	**(0.011)**	(0.011)	(0.013)	(0.011)	(0.011)	(0.32)	(0.18)	(0.35)	**(0.18)**	(0.29)	(0.022)	(0.022)	(0.020)	**(0.017)**	(0.016)

cess (ANHP) [2]. Both THP and ANHP are transformer-based models that utilize the full set of observed events $\mathcal{H}_{t_n}$ to predict e_{n+1}. IFL directly models $f^*(t,k) = f^*(t)f^*(k|t)$ using a log-normal mixture for a more flexible distribution. As it explicitly parameterizes the distribution $f^*(t)$, the expected value $\mathbb{E}_{f^*}[t]$ can be computed analytically. Additional implementation details are provided in Sec.5.6.3.

5.1.2 Comparison of Results

We evaluate the performance of linear Dec-ODE against state-of-the-art TPP methods using the five real-world datasets described in Sec. 5.1.1. A summary of the results is provided in Table 5.1. Overall, our approach achieved results that were comparable to or better than the baselines across all metrics. This demonstrates that Dec-ODE effectively captures the complex dynamics of MTPPs through independently propagated influence functions. Notably, Dec-ODE excelled in prediction tasks, as evidenced by its low RMSE, while ANHP slightly outperformed it in NLL. Additionally, methods like Dec-ODE and IFL, which estimate $f^*(t)$ to compute $\mathbb{E}[t]$, tended to predict event times more accurately.

Discussion. To ensure a fair comparison, we increased the number of samples used in the thinning algorithm [30, 31] implemented by [2]. In most cases, the sample count was increased by a factor of 20. However, on the Reddit dataset with THP, the thinning algorithm struggled to sample correctly from $f^*(t)$ due to the low magnitude and high fluctuation of the intensity function. Additional details are provided in Sec.5.6.4.

5.1.3 Explainability of Dec-ODE

In MTPP and TPP, events have a temporal and complex influence on $\lambda^*(t)$ and $f^*(t)$ [10, 22]. Thus, when evaluating the explainability of a neural network modeling an MTPP, it is crucial to understand how individual events impact predictions and how these effects evolve over time. Dec-ODE inherently provides this information. For instance, the temporal behavior of $\lambda_g^*(t)$ and $\hat{f}^*(k|t)$ can be directly visualized, as shown in Fig. 5.1.

Our detailed analysis of the Retweet dataset further corroborates Dec-ODE's explainability. For example, meaningful patterns in the Retweet data are visualized in Fig. 5.2. The dataset comprises three classes: a post by a user with few followers ($\sim 50\%$ of the population), a post by a user with a moderate number of followers ($\sim 45\%$), and a post by a user with many followers ($\sim 5\%$), labeled as 0, 1, and 2, respectively. In Fig. 5.2(a), which visualizes $\mu(t; e_i)$ conditioned on different e_i, each event shows a temporally decaying influence on $\lambda_g(t)$. This aligns with the expected behavior of social media posts, which generally elicit immediate reactions rather

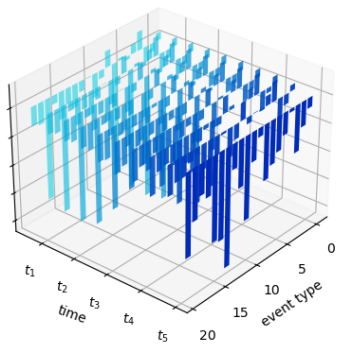
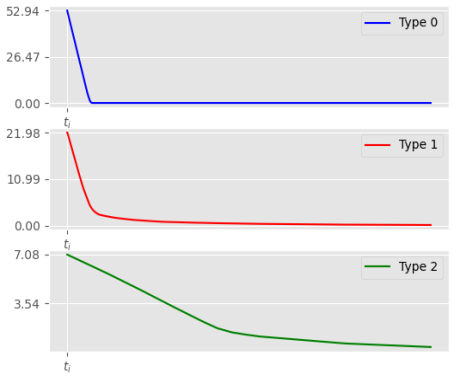
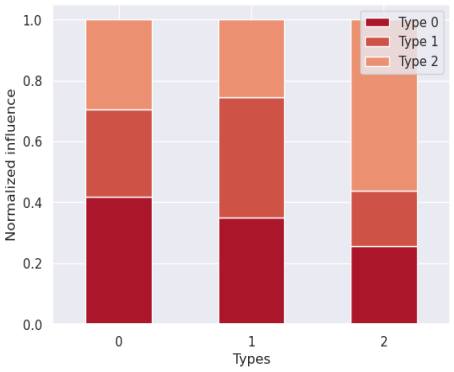


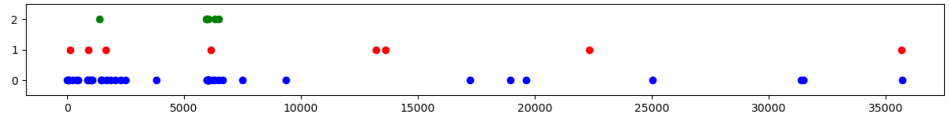
Figure 5.1: Visualization of propagated $\hat{f}^*(k|t, e_i)$ in the StackOverflow experiment. The axes represent time, event type, and magnitude, respectively.



(a) Influence function $\mu(t)$ conditioned on different event types, plotted over the same time range.



(b) Averaged influence proportions between different marks.



(c) Visualization of event marks over time. Blue (0): users with few followers, Red (1): users with moderate followers, Green (2): users with many followers.

Figure 5.2: Visualization of patterns in the Retweet dataset. (a) $\mu(t; e_i)$ for different marks, (b) Interaction between marks, (c) Event marks over time.

than delayed effects. Moreover, posts from users with many followers exhibit slower decay, as they tend to have broader exposure and sustain their influence for a longer time.

Furthermore, the averaged proportions of influence across different mark types are shown in Fig. 5.2(b). First, despite type 2 accounting for only 5% of the population, it exerts significant influence on the overall occurrence of events. This is consistent with the notion that individuals with many followers tend to impact a larger audience. Second, events of the same type have the strongest mutual influence. Given the exponentially decaying pattern of $\mu(t)$ and the large initial influence magnitude of type 0 (Fig. 5.2(a)), events of type 0 likely exhibit strong clustering. This inference is validated by the visualization in Fig. 5.2(c).

5.1.4 Parallel Computing Scheme

While Neural ODEs are highly effective at learning the continuous dynamics of a system, propagating the hidden state requires solving the IVP step by step, which can be computationally intensive. In Sec. 3.2.3, we introduced a method to propagate the hidden state through time in parallel by disentangling individual events. We validate this optimization scheme by comparing the average iteration time with that of traditional IVP-based sequential propagation.

For the sequential approach, the differential equation is solved incrementally from t_0 to t_n. Table 5.2 presents the results, demonstrating a significant reduction in computation time across all cases. The parallelized computation for Dec-ODE was at least twice as fast (e.g., for MIMIC-II) and up to five times faster (e.g., for Reddit) compared to the traditional sequential scheme.

Table 5.2: Training efficiency comparison in different datasets with the average time taken for an iteration (sec/iter) as the metric. Ratio shows that the iteration time at least reduces in half.

StackOverflow			MIMIC-II			Retweet			Reddit		
parallel	Sequential	Ratio	parallel	Sequential	Ratio	parallel	Sequential	Ratio	parallel	Sequential	Ratio
15.0	57.7	0.26	2.9	6.5	0.47	16.8	67.6	0.25	15.5	78.7	0.20

5.2 Linear vs. Non-Linear Influence Function

In Table 5.1, the basic Dec-ODE model with a linear Φ_λ was evaluated. However, both Φ_λ and Φ_k can be defined as any functions, including neural networks, as long as they meet the conditions outlined in Secs. 3.1.2 and 3.1.4. Notably, with a linear Φ_λ, every influence function must satisfy $\mu(t) \geq 0$ to ensure the intensity remains positive. This constraint means that only excitatory effects, where events increase the intensity, can be modeled.

In this section, we explore the extensibility of our framework by introducing and

testing a less restricted variant, referred to as the *non-linear* Φ_λ. The non-linear $\Phi_\lambda(t)$ is defined as follows:

$$\Phi_\lambda(\mu(t;e_0),\cdots,\mu(t;e_i)) = \text{softplus}\Big(\sum_{e_i\in\mathcal{H}_t}\mu(t;e_i)\Big). \tag{5.1}$$

Through this modeling approach, inhibitory effects, where an event reduces the overall intensity, can be incorporated.

To assess how this relaxation influences the modeling of TPP, both linear and non-linear models are tested in a simplified setting, using only the first 40 events from each sequence. The experimental results are summarized in Table 5.3, with NLL as the evaluation metric. In most cases, the overall result regarding the ground intensity improved, indicating that a more flexible Φ_λ allows for a more accurate prediction of the intensity function.

However, the non-linear Dec-ODE model reveals limitations when predicting time-delayed effects, where an event does not immediately influence the intensity but instead has a delayed effect. This issue arises because the inhibitory effect outweighs the excitatory one, causing the intensity function to drop to zero. In such situations, the inhibitory effect hinders the model's performance.

In conclusion, this experiment suggests that a more flexible Φ_λ can lead to higher fidelity predictions in most cases. However, further discussion is needed regarding the introduction of constraints to more effectively model various patterns in TPP.

Table 5.3: Experiment on non-linear Φ_λ. The softplus is applied after the summation in order to express inhibitory effects.

StackOverflow		MIMIC-II		Retweet		Reddit	
Linear	Non-linear	Linear	Non-linear	Linear	Non-linear	Linear	Non-linear
0.9948	**0.8987**	0.2451	**0.2183**	−5.0872	**-5.1649**	**0.5163**	18.2571

5.3 Imputation

In this section, we explore the impact of independently modeling hidden state dynamics by comparing it with methods that utilize contextual information. To examine this, events from the StackOverflow dataset are randomly removed to observe how the behavior changes as the number of observed events decreases.

For a fair comparison, we randomly selected 90%, 80%, 70%, and 60% of the indexes from the test dataset, and saved the corresponding data as new test sets. The results using these new test sets are visualized in Figure 5.3. The graph shows that the performance decline of Dec-ODE follows a similar trend when compared to other methods.

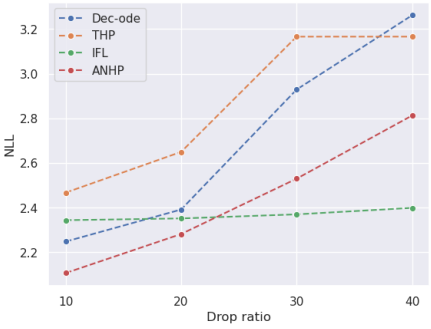


Figure 5.3: Imputation experiment done using Stackoverflow dataset, where from 10% to 40% of data are randomly dropped.

This suggests that the other methods are unable to recover from the loss of information. This conclusion is supported by the fact that Dec-ODE, which does not account for inter-event relationships, exhibits a similar trend.

5.4 Simulation Study

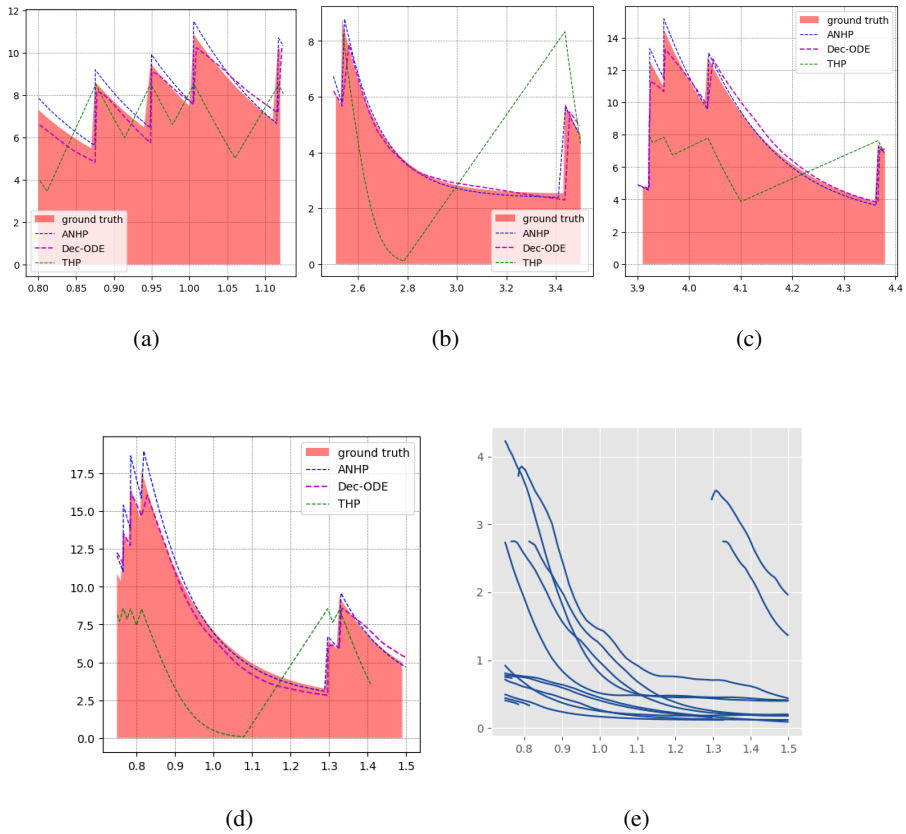


Figure 5.4: Visualization from a simulation study. (a), (b), (c), and (d) compare the true intensity function of a Hawkes process and the reconstructed results from THP, ANHP, and Dec-ODEs. Each $\mu(t)$ that composes the result in (d) is illustrated in (e).

We conducted a simulation study on the Hawkes process, following a similar procedure to that of the Neural Hawkes Process (NHP) [12]. To obtain the true intensity function, we randomly sampled parameters for the kernel function of the multivariate Hawkes process and simulated synthetic data using the tick library [32]. The sampling range was adjusted from the NHP due to differences in scale between the library and the original paper. In Figures 5.4(a) - 5.4(d), THP struggles to capture the flexible dynamics of the intensity function λ_g, while ANHP and Dec-ODE exhibit dynamics

	RMSE	ACC	NLL
THP	0.7734	27.48	3.0121
ANHP	0.6528	30.31	0.2807
Dec-ODE	**0.6568**	**30.35**	**0.2739**

Table 5.4: Estimation accuracy on the simulated Hawkes process.

closely resembling the ground truth intensity. Table 5.4 further shows that Dec-ODE outperforms both THP and ANHP across all metrics. These results align with those in Table 5.1, suggesting that Dec-ODE is capable of reliably simulating MTPPs compared to other state-of-the-art methods. Figure 5.4(e) visualizes $\mu(t)$ components that make up $\lambda_g(t)$ from Figure 5.4(d), demonstrating that the ground intensity $\lambda_g(t)$ can be reconstructed from the individually modeled trajectories, i.e., an MTPP can be effectively modeled using decoupled information.

5.5 Train time comparison

The time required for training one epoch (sec / epoch) is measured in order to compare the time required for training. The experiment is conducted using a single NVIDIA RTX A6000 GPU (48GB memory) with the largest batch size possible. The summary of the expriment can be found in the table 5.5. THP required the least amount of time, while methods predicting continuous dynamics of intensity showed a relatively slower training rate. In most cases, Dec-ODE shows a shorter training time per epoch.

Table 5.5: Training time (sec / epoch) compared between THP, ANHP, and Dec-ODE.

	MOOC	Reddit	Retweet	Stackoverflow	MIMIC-II
THP	12.94	66.95	34.05	13.36	2.27
ANHP	835.36	541.20	630.64	129.51	3.69
Dec-ODE	117.35	242.75	484.08	123.28	8.12

Table 5.6: Required memory (MB) compared between baseline methods with batch size 4.

	MOOC	Reddit	Retweet	Stackoverflow	MIMIC-II
RMTPP	$1110 + 1114$	$1706 + 1142$	$1294 + 1112$	$1306 + 1114$	$1114 + 1106$
THP	3484	15304	1678	3808	1130
ANHP	31310	< 48 GB	11790	39260	1244
Dec-ODE	1422	1472	1314	1422	1470

5.6 Implementation details

5.6.1 IVP solving with varying time intervals

The Initial Value Problem (IVP) solving with varying time intervals is performed following the approach outlined in [15]. In brief, the integration over the region $[t_{start}, t_{end}]$ is solved within the normalized interval $[0, 1]$, and then rescaled back to the original time range. This method allows for the computation of time intervals of different lengths with the same number of steps. We apply this technique to both batch computations with varying lengths and the parallel computation of $h(t; e_i)$ as discussed in Sec. 3.2.3.

5.6.2 Batch Computation

Our method extends the propagation of $h(t)$ beyond t_N, the last observed time point. To fully cover the range of $f^*(t)$, two distinct masking operations are necessary. The first is the commonly used *sequence mask*, which is applied when dealing with sequences of varying lengths. For sequences with different lengths, unobserved events must be ignored during both training and inference, and the sequence mask serves to mask out these unobserved time points. The second mask is the *propagation mask*, which ensures that the unobserved time points after t_N are not masked. This mask is used when solving ODEs to propagate $h(t)$ until the decoded $\mu(t)$ reaches convergence.

5.6.3 Baseline

For THP and ANHP, we utilized the public GitHub repository ([2] with MIT License). The code for THP has been modified by [2], where time and event prediction is now performed using $\lambda(t)$ with a thinning algorithm, as opposed to the neural network-based prediction module from the original version.

Additionally, as noted in the public GitHub repository of THP, there is an issue with the NLL computation. In both published versions, NLL is computed conditioned on the observed history and the current event, i.e., $L(e_i) = f(e_i|e_0, \ldots, e_i)$. Therefore, the intensity calculation has been corrected to $L(e_i) = f(e_i|e_0, \ldots, e_{i-1})$, based on the integration code modifications.

For IFL, we used the public GitHub repository https://github.com/shchur/ifl-tpp ([24], no license specified). Several modifications were necessary, as the original implementation does not support time point prediction. These changes were based on the author's comments in the repository's "issue" page. However, the computation of $\mathbb{E}[t]$ was unstable due to the high variance in the mixture model components. Specifically, when a distribution in the mixture model has high variance, the exponential term in the calculation causes an overflow, leading to an overall failure of the computation. To address this, we tightened the gradient clipping parameters in [24], as recommended in [26].

5.6.4 Thinning algorithm

In the experiment presented in Table 5.1, we tested various parameters for the thinning algorithm [30, 31], as implemented by [2], to ensure a fair comparison. THP and ANHP use the thinning algorithm to sample the next time points and then compute $\mathbf{E}[t]$ by averaging these samples.

The thinning algorithm operates similarly to rejection sampling, where a proposal is accepted with the probability $\lambda(t)/\lambda_{up}$, with the condition that $\lambda_{up} \geq \lambda(t)$ for all $t \in (t_{i-1}, \infty)$ [12]. Thus, a reliable value for λ_{up} is crucial to sample from the correct

distribution. If λ_{up} is set too high, all samples will be rejected, whereas if it is set too low, incorrect samples will be accepted.

In the implementation, λ_{up} is calculated as $c \times \max(\lambda(s_0), \ldots, \lambda(s_m))$, where c is a constant, m is the number of samples used for the calculation, and s_m is a uniformly sampled time point. To obtain an accurate λ_{up}, for most of the reported results in Table 5.1, we increased m by a factor of 10. In cases where the overall $\lambda(t)$ is very low, the scale of λ_{up} was increased up to 1000 in extreme cases.

5.6.5 Training Details

When tuning the simple Dec-ODE, three hyperparameters were considered for the model structure, alongside the Initial Value Problem (IVP) solving method. The hyperparameters are the dimension of the hidden state D, the dimension of the linear layers of the neural network N, and the number of linear layers L. We did not conduct an exhaustive search for the optimal parameters for each dataset; instead, we applied similar parameters across all datasets for testing.

Throughout the experiment, D was selected from $\{32, 64\}$. Since the dynamics of $h(t)$ heavily depend on the information in the hidden state, we anticipate that a higher dimension allows the neural network to capture more complex dynamics.

The width of the linear layers N was chosen from $\{128, 256\}$, and the number of layers L was tested from $\{3, 4, 5\}$. In most cases, varying the number of layers did not yield significant improvements in performance.

The best-performing hyperparameters were selected based on the results, with $D = 64$, $N = 256$, and $L = 3$ being used most frequently.

For the IVP solver, two different methods were used for training and testing. During training, Euler's method was primarily used for efficiency. Since the goal of training is to learn the dynamics of the hidden state $h(t)$, we believe that Euler's method sufficiently meets this purpose, though we did not extensively explore the benefits of other solvers.

For testing $\lambda_g(t)$, we used the more accurate Runge-Kutta method with a fixed step size, commonly referred to as RK4. The RK4 method computes the solution as follows:

$$y_{n+1} = y_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) \tag{5.2}$$

$$t_{n+1} = t_n + h \tag{5.3}$$

where,

$$k_1 = f(t_n, y_n), \tag{5.4}$$

$$k_2 = f(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_1), \tag{5.5}$$

$$k_3 = f(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_2), \tag{5.6}$$

$$k_4 = f(t_n + h, y_n + hk_3). \tag{5.7}$$

The RK4 method provides a more precise trajectory than Euler's method, reducing the error in the estimated $f^*(t)$. Other methods, such as dopri5, RK4 with step-size control, and other IVP solver variants, can be applied for even more accurate results. However, applicaiton of such methods were not fully explored in this study.

For solving the IVP, we fixed the number of steps required to solve the interval from t_i to t_{i+1}. Similar to the IVP solver, a simpler setting was used for training and a more precise setting for testing. During training, 16 steps were used between each event. Although increasing the number of steps is expected to improve results, we did not explore this in detail since it produced comparable results to state-of-the-art methods. For testing, the number of steps was increased to 64.

When testing $f(k|t)$, the precise approximation of $\hat{f}(k|t, e_i)$ had no noticeable effect on the results. Therefore, for computational efficiency, Euler's method was used with the same number of steps as used during training.

Datasets	# of Seq.	# of Events	Max Seq. Length	# of Marks
MOOC	7,047	389,407	200	97
Reddit	10,000	532,026	100	984
Retweet	24,000	21173,533	264	3
StackOverflow	6,633	480,414	736	22
MIMIC-II	650	2419	33	75

Table 5.7: Statistics of benchmark datasets used for comparisons.

5.7 Dataset Description

5.7.1 Benchmark dataset

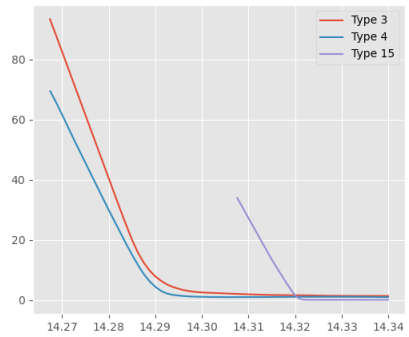
MIMIC-II and Retweet were from the public GitHub repository ([1], with no license specified). Others were also from the public GihtHub repository ([25], with no license specified).

5.7.2 Data preprocess

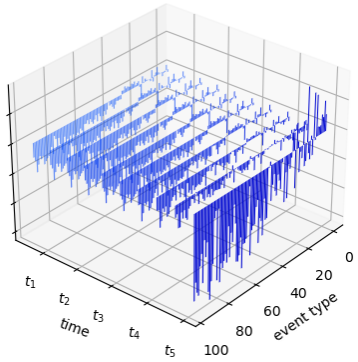
When working with the Mooc, Reddit, Retweet, and StackOverflow datasets, two modifications were made. First, when two events have the same time point, one of them is removed, specifically the latter event in the dataset. This adjustment was made because when $t_i - t_{i-1} = 0$, the IFL method was unable to properly estimate $f^*(t)$. Additionally, in many cases, TPP assumes that two or more events occurring at the same time is highly improbable.

Second, as mentioned in Sec. 5.1.1, the data were scaled by the standard deviation of $\{t_i - t_{i-1}\}_{i=0}^{n}$. When the scale of $t_i - t_{i-1}$ is excessively large, it can cause overflow during computations. To mitigate this, we aimed to match the scale to the results from [26], although the exact scaling factor was not specified in the original work, so we chose the standard deviation as the scaling method.

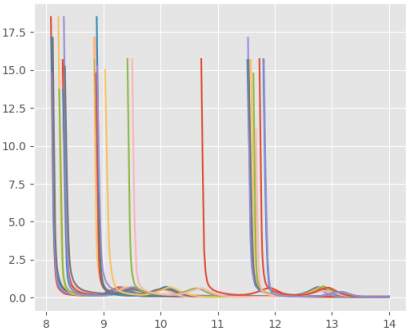
5.8 Visualizations



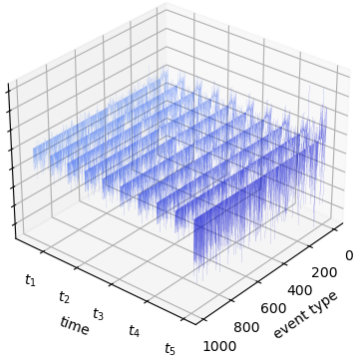
(a) Visualization of $\mu(t; e_i)$ trained using MOOC.



(b) $\hat{f}(t; e_i)$ trained using in MOOC.

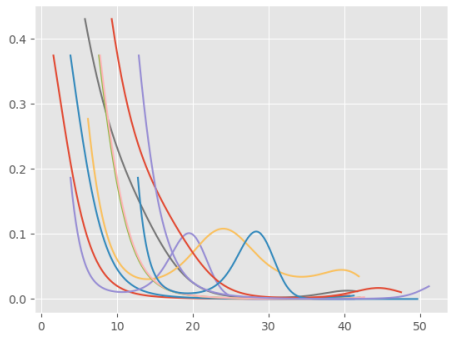


(c) $\mu(t; e_i)$ trained using Reddit.

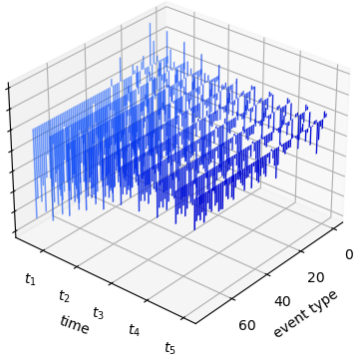


(d) $\hat{f}(t; e_i)$ trained on Reddit dataset.

Figure 5.5: Visualization of dynamics of μ and $\hat{f}$ in benchmark dataset. (a) is a visualization of ground intensity trained on MOOC dataset, (b) is a visualization of $\hat{f}(t; e_i)$ changing through time trained using MOOC dataset, (c) is a visualization of ground intensity trained on Reddit dataset, and (d) is a visualization of $\hat{f}(t; e_i)$ changing through time trained using Reddit dataset.



(a) Visualization of $\mu(t; e_i)$ trained using SO.



(b) Visualization of $\hat{f}(t; e_i)$ in MIMIC.

Figure 5.5: Visualization of dynamics of μ and $\hat{f}$ in benchmark dataset. (a) shows dynamics of ground intensity trained using Stack Overflow dataset, and (b) shows $\hat{f}(t; e_i)$ changing through time trained using MIMIC-II dataset.

VI. Conclusion

In this work, we introduced a novel framework for modeling MTPPs, referred to as Dec-ODE, which decouples the influence of each event to enhance flexibility and computational efficiency. The Dec-ODE utilizes neural ODEs to model the continuously evolving influence of individual events, which control the overall dynamics of the MTPP. Additionally, we proposed a training scheme for a linear Dec-ODE, enabling the efficient parallel computation of influences. The method was validated on five real-world benchmark datasets, demonstrating comparable or superior performance. Furthermore, it offers explainability in the modeling process, highlighting its potential for applications in areas such as out-of-distribution detection and survival analysis.

요 약 문

사건의 종류와 발생 시간의 쌍으로 이루어진 데이터는 의료, 산업, 금융 등 다양한 분야에서 수집되고 사용되고있다. 이러한 시계열 데이터를 확률론적 과정으로 모델링을 한다면 정보를 분석하거나 미래에 발생할 정보들을 예측하는 것에 효과적으로 사용될 수 있다. 특히 사건의 발생 시간에 대한 예측을 위해 자주 사용되는 방법론으로는 Marked Temporal Point Process (MTPP) 가 있다.

이러한 MTPP를 인공지능을 활용하여 모델링하는 것에는 다양한 접근 방법이 제안되었다. 이중에서 대표적인 접근 방법들로는 RNN과 Transformer기반의 모델을 사용하는 방법들이 성능적인 측면에서 효과를 입증하였다. 특히 이러한 방법론들은 사건들 사이의 관계성을 학습하여 효과적인 예측을 하는 것을 장점으로하지만, 모델이 어떠한 이유로 이러한 결과를 출력하였는지에 대한 설명성을 잃는 경우가 많다. 하지만 실제 생활에서는 다음 사건의 예측 뿐 아니라, 영향력에 대한 분석, 생존분석, 그리고 사건의 확률 분포에 대한 이해 등 다양한 활용이 요구된다. 다양한 분야에 활용되기 사건의 개별적인 영향력을 파악하는 것은 큰 중요성을 갖는다.

이러한 문제점을 해결하고자 본 논문은 이벤트들의 개별적인 영향력을 표현하는 프레임워크를 제안하였다. 각 영향력은 상미분방정식을 통해 모델링되고, MTPP에 대한 특성을 미분방정식의 특성을 활용하여 효율적으로 계산을 하는 방법론 또한 함께 제안하였다. 마지막으로 제안된 프레임워크의 효과성을 입증하기 위하여 단순한 구조를 갖는 모델과, 해당 모델을 활용한 더 효율적인 학습 방법을 제안하였다.

이벤트의 영향력을 설명가능하게 표현하기 위해 각각의 이벤트는 독립적인 시

간에 따라 변화하는 프로세스로 표현된다. 이때 보다 효율적인 계산을 위해 다차원 프로세스를 상미분방적식으로 표현한다. 이 프로세스는 이벤트의 발생 시간에 대한 확률 분포를 계산하기 위한 μ와 이벤트의 종류의 분포를 계산하기 위한 $\hat{f}$로 변환되게 된다. 시간에 대한 효율적인 예측을 위해 확률분포, 생존률 등의 예측값들의 시간에 따른 적분 값들은 미분방정식을 활용하여 프로세스와 함께 계산한다. 이렇게 독립적으로 모델링을 하는 것은 예측을 하는 것에 원하는 이벤트의 집합을 고려할 수 있는 장점을 제공한다.

이러한 방법론의 효과성을 보이기 위해 과거의 영향을 선형 합산 등 단순한 함수를 통해 관심 시점의 값을 계산하는 구현을 제안하였다. 이렇게 단순한 구현하는 것의 장점으로는 학습시 적분을 위해 모든 시작 시점부터 순차적으로 계산을 하는 것이 아니라, 각각의 이벤트를 자신의 시점에 맞추어 동시에 계산 후 추후 손실함수 계산을 위해 사용 될 수 있다.

해당 프레임워크는 설명가능하고 연속적인 MTPP의 모델링을 하는 것 뿐 아니라, 최고 성능을 내는 기존 방법론보다 좋거나 유사한 성능을 보이는 것을 보였다. 또한 제안된 방법론의 효율성 및, 프레임워크의 확장성이 실험을 통해 확인 되었다.

References

[1] Simiao Zuo, Haoming Jiang, Zichong Li, Tuo Zhao, and Hongyuan Zha. Transformer Hawkes Process. In *International Conference on Machine Learning*, 2020.

[2] Chenghao Yang, Hongyuan Mei, and Jason Eisner. Transformer embeddings of irregularly spaced events and their participants. In *International Conference on Learning Representations*, 2022.

[3] Qingyuan Zhao, Murat A. Erdogdu, Hera Y. He, Anand Rajaraman, and Jure Leskovec. Seismic: A self-exciting point process model for predicting tweet popularity. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2015.

[4] Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection, 2014.

[5] Nan Du, Hanjun Dai, Rakshit Trivedi, Utkarsh Upadhyay, Manuel Gomez-Rodriguez, and Le Song. Recurrent marked temporal point processes: Embedding event history to vector. In *Special Interest Group on Knowledge Discovery and Data Mining*, 2016.

[6] Rui Meng, Fan Yang, and Won Hwa Kim. Dynamic covariance estimation via predictive wishart process with an application on brain connectivity estimation. *Computational Statistics & Data Analysis*, 185:107763, 2023.

[7] Emmanuel Bacry, Adrianna Iuga, Matthieu Lasnier, and Charles-Albert Lehalle. Market impacts and the life cycle of investors orders. *Market Microstructure and Liquidity*, 1(02), 2015.

[8] Alan G. Hawkes. Hawkes processes and their applications to finance: a review. *Quantitative Finance*, 18(2), 2018.

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

[10] Alan G. Hawkes. Spectra of some self-exciting and mutually exciting point processes. *Biometrika*, 58(1), 1971.

[11] L Adamopoulos. Cluster models for earthquakes: Regional comparisons. *Mathematical Geology 8*, 1976.

[12] Hongyuan Mei and Jason Eisner. The neural Hawkes Process: A neurally self-modulating multivariate point process. In *Neural Information Processing Systems*, 2017.

[13] Junteng Jia and Austin R. Benson. Neural jump stochastic differential equations. In *Neural Information Processing Systems*, 2019.

[14] Qiang Zhang, Aldo Lipani, Omer Kirnap, and Emine Yilmaz. Self-attentive Hawkes Process. In *International Conference on Machine Learning*, 2020.

[15] Ricky T. Q. Chen, Brandon Amos, and Maximilian Nickel. Neural spatio-temporal point processes. In *International Conference on Learning Representations*, 2021.

[16] Takahiro Omi, Naonori Ueda, and Kazuyuki Aihara. Fully neural network based model for general temporal point processes. In *Neural Information Processing Systems*, 2019.

[17] A. De, U. Upadhyay, and M. Gomez-Rodriguez. Temporal point processes, 2019.

[18] D.J. Daley and D. Vere-Jones. *An Introduction to the Theory of Point Processes*, volume 1. Springer, 2003.

[19] Renqin Cai, Xueying Bai, Zhenrui Wang, Yuling Shi, Parikshit Sondhi, and Hongning Wang. Modeling sequential online interactive behaviors with temporal point process. In *International Conference on Information and Knowledge Management*, 2018.

[20] Nan Du, Mehrdad Farajtabar, Amr Ahmed, Alexander J. Smola, and Le Song. Dirichlet-hawkes processes with applications to clustering continuous-time document streams. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2015.

[21] Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations. In *Neural Information Processing Systems*, 2018.

[22] Valerie Isham and Mark Westcott. A self-correcting point process. *Stochastic Processes and their Applications*, 8(3), 1979.

[23] Zhimeng Pan, Zheng Wang, Jeff M. Phillips, and Shandian Zhe. Self-adaptable point process with nonparametric time decays. In *Neural Information Processing Systems*, 2021.

[24] Oleksandr Shchur, Marin Biloš, and Stephan Günnemann. Intensity-free learning of temporal point processes. In *International Conference on Learning Representations*, 2020.

[25] Haitao Lin, Lirong Wu, Guojiang Zhao, Pai Liu, and Stan Z. Li. Exploring generative neural temporal point process. In *Transactions on Machine Learning Research*, 2022.

[26] Wonho Bae, Mohamed Osama Ahmed, Frederick Tung, and Gabriel L. Oliveira. Meta temporal point processes. In *International Conference on Learning Representations*, 2023.

[27] Ming Jin, Yuan-Fang Li, and Shirui Pan. Neural temporal walks: Motif-aware representation learning on continuous-time dynamic graphs. In *Neural Information Processing Systems*, volume 35, 2022.

[28] Nabeel Seedat, Fergus Imrie, Alexis Bellot, Zhaozhi Qian, and Mihaela van der Schaar. Continuous-time modeling of counterfactual outcomes using neural controlled differential equations. In *International Conference on Machine Learning*, 2022.

[29] Sungwoo Park, Byoungwoo Park, Moontae Lee, and Changhee Lee. Neural stochastic differential games for time-series analysis. In *International Conference on Machine Learning*, 2023.

[30] Yuanda Chen. Thinning algorithms for simulating point processes, 2016.

[31] Y. Ogata. On lewis' simulation method for point processes. *IEEE Transactions on Information Theory*, 27(1), 1981.

[32] Emmanuel Bacry, Martin Bompaire, Stéphane Gaïffas, and Soren Poulsen. Tick: a python library for statistical learning, with a particular emphasis on time-dependent modelling, 2018.

Acknowledgements

새로운 도전을 할 수 있게 장려해주신 김원화 교수님 감사합니다. 교수님의 지도를 통해 항상 새로운 경험을하고 성장하기 위해 포기하지 않고 도전 할 수 있었습니다. 또한 연구실 동료들에게도 함께하며 많은 도움을 받았기에 감사드립니다.

현아 - 꾸준히 연구하는 모습을 보며 저 또한 나태해지지 않고 많이 배울 수 있었습니다. 연구실에 대해 자세히 알려준 덕분에 합류할 수 있었습니다, 감사합니다. 지은 - 항상 긍정적인 에너지를 주는 모습에 연구실에 잘 적응 할 수 있었습니다. 승훈 - 매번 맡은일 현명하게 해결하는 모습을 보며 많이 배울 수 있었습니다. 준혁 - 묵묵히 정진하는 모습, 그리고 스스로를 고찰하며 발전해 나아가는 모습을 보며 많이 배울 수 있었습니다. 유빈 - 항상 주변을 잘 챙기며 꼼꼼하게 능력이 정말 멋있는 선배였습니다. 재윤 - 못하는 것 없고, 꾸준히 본인 할 일 해나아가는 모습 정말 멋있다고 생각했습니다. 수연 - 정말 멋있는 연구 함께 해준 수연누나 감사합니다. 예찬 - 넓은 시야를 갖고있는 멋있는 동네 주민이었습니다, 주변사람들을 배려하고 배우는 자세를 보며 많이 배웠습니다. 민재 - 의도한 것 같지는 않지만 주변을 즐겁게 해준 덕분에 행복하게 연구실 생활을 할 수 있었습니다. 연구를 스스로의 관점으로 이해하고자 하는 모습을 보며 많이 배웠습니다. 성우 - 연구하는 것에 있어 가장 많은 디스커션을 해준 성우형 감사합니다. 이론적인 베이스를 위해 끊임 없이 공부하고 공부하는 모습에 정말 큰 자극을 많이 받았습니다. 제가 추구하는 연구의 방향성에도 항상 영향을 받았습니다. 세형 - 연구를 대하는 자세를 보며 많이 배웠습니다. 함께 연구를 하지 못해서 정말 아쉬움이 크게 남지만 멀리서 응원하겠습니다. 동현 - 저의 첫 연구를 함께 해주어서 정말 감사했습니다. 다재다능한 능력에 정말 큰 도움이 되었고, 새로운 것을 배우고자 하는 모습 또한 큰 귀감이 되었습니다. 성윤 - 문제 정의를 하는 모습, 질문을 하는 모습을 보며 많이 감탄하였습니다. 정말 능력이 뛰어난 선배같은 동료였습니다. 민재 - 긍정적인 태도와 밝은 모습으로 IPIU 룸메부터 졸업까지 큰 도움을 받았습니다. 하영 - 씩씩하게 새로운 것을 배우고자 하는 태도, 긍정적인 에너지를 통해 새로운 도전을 하는 것에 큰 도움을 받았습니다. 수진

- 성실하고 쾌활하게 연구하는 모습이 정말 인상 깊었습니다. 좋은 연구 많이 하길 응원합니다. 승주 - 새로운 주제를 공부하고 함께 의견을 나누며 흥미로운 주제를 고민하고 알아갈 수 있었습니다, 감사합니다. 재진 - 더 많은 대화를 나누고 교류하고 싶었지만 한학기 밖에 함께하지 못해 정말 아쉽습니다.

Curriculum Vitae

Name : Yujee Song

Education

2023. 02. – 2025. 02. Graduate School of Artificial Intelligence, POSTECH (M.S.)

2020. 03. – 2022. 02. Computer Science & Engineering, Chung-Ang University (B.S.)

2015. 09. – 2017. 06. Computer Science & Engineering, UC Irvine (B.S.)

Experience

2024. 07. – 2024. 09. Medical AI Research & Development Intern, VUNO

2023. 02. – 2025. 02. Medical Information Processing Lab, Graduate School of Artificial Intelligence, POSTECH