A Proofs

A.1 Proof of Theorem 1

Proof. Denote that the jump times of the multivariate Hawkes process $\mathbf{N}(t) = (N_1(t), \dots, N_V(t))^{\mathrm{T}}$, and the kernel function $g(t) = \exp(-\beta t)$. On the time interval between jumps $(t_{i-1}, t_i]$, the intensity function $\lambda_k(t)$ for each type $k \in \mathcal{V}$ behaves as an ODE as follows.

$$\begin{aligned}\frac{\mathrm{d}\lambda_k(t)}{\mathrm{d}t} &= \sum_{j=1}^{V} \sum_{(v',t')\in\mathcal{H}(t),v'=j} \alpha_{jk}(-\beta\exp(-\beta(t-t'))),\\ \mathrm{d}\lambda_k(t) &= -\beta\sum_{j=1}^{V} \sum_{(v',t')\in\mathcal{H}(t),v'=j} \alpha_{jk}(\exp(-\beta(t-t_i)))\mathrm{d}t,\\ &= -\beta(\lambda_k(t)-\mu_k)\mathrm{d}t = \beta(\mu_k - \lambda_k(t))\mathrm{d}t.\end{aligned} \tag{1}$$

When an event j occurs at t_i, then the jump size of $\lambda_k(t_i)$ is given by $\Delta\lambda_k(t_i) = \alpha_{jk}$. The right-limit at t_i is $\lambda_k(t_i^+) = \lambda_k(t_i) + \alpha_{jk}$. When considering all possible event types, the jump size $\Delta\lambda_k(t_i) = \sum_{j=1}^{V} \alpha_{jk}\Delta N_j(t_i)$. Therefore, for the Hawkes process with an exponential kernel function, it has an ODE jump function as Eq. (7) and can be further extended to the SDEs. □

A.2 Proof of Theorem 2

We mainly proof the Theorem 2 for one proposed SDE variant as Eq. (8), and it can be easily adapted to the neraul LNSDE variant as Eq. (9).

Proof. Let us focus on one event type k, since the SDEs of different type are independent between jump times $(t_{i-1}, t_i]$. Consider a probability space $(\Omega, \mathcal{F}, \mathbb{P})$ on which we define a d_Z-dimensional Brownian motion $W_k = \{W_k(t)\}_{t\geq 0}$ for each type k, and a counting process $\mathbf{N} = \{\mathbf{N}(t)\}_{t\geq 0}$ that jumps at the times $\{t_i\}_{i=1}^{\infty}$, where $\mathbf{N}(t) = (N_1(t), \dots, N_V(t))^{\mathrm{T}}$. Suppose that W_k and $\mathbf{N}$ are independent. Then, define the filtration $\{F_t\}_{t\geq 0}$ as the augmented natural filtration of W_k and $\mathbf{N}$, for all $t \geq 0$, Hence, $(\Omega, \mathcal{F}, \{\mathcal{F}_t\}_{t\geq 0}, \mathbb{P})$ is a filtered probability space that satisfies the usual conditions. We define the stochastic process $\mathbf{Z}_k = \{Z_k(t)\}_{t\geq 0}$ and as the solution of the SDE for event type k. For all event types, we define $\mathbf{Z}(t) = (Z_1(t), \dots, Z_V(t))^{\mathrm{T}}$.

Firstly, we show the existence and uniqueness of solutions $\mathbf{Z}_k$ in the time interval $[0, t_1]$. For an event type k, it follows the SDE as Eq. (8).

$$\begin{cases}\mathrm{d}Z_k(t) = f(Z_k(t), t_{i-1}, t-t_{i-1})\mathrm{d}t + g(Z_k(t))\mathrm{d}W_k(t),\\ Z_k(0) = z_k,\end{cases}$$

where $z_k \in \mathbb{R}^{d_Z}$ is the starting point and the measurable functions $f : \mathbb{R}^{d_Z} \times \mathbb{R}_+ \times \mathbb{R}_+ \to \mathbb{R}^{d_Z}$ and $g : \mathbb{R}^{d_Z} \to \mathbb{R}^{d_Z}$ are the drift function and the diffusion function, $d_Z \in \mathbb{N}$. Then, we impose the following assumptions as discussed in [Oh *et al.*, 2024]:

- For any neural network $s(x, t, t; \theta_s) : \mathbb{R}^{d_1} \times \mathbb{R}_+ \times \mathbb{R}_+ \to \mathbb{R}^{d_2}$ with parameter θ_s, there exists a positive constant $L_s > 0$ such that for all $t \geq 0, x, x' \in \mathbb{R}^{d_1}$.
$$|s(x,t,t;\theta_s) - s(x',t,t;\theta_s)| \leq L_s|x - x'|$$
- For any neural network $s(x, t, t; \theta_s) : \mathbb{R}^{d_1} \times \mathbb{R}_+ \times \mathbb{R}_+ \to \mathbb{R}^{d_2}$ with parameter θ_s where the last layer is the `ReLU` or linear function, there exists a positive constant $K_s > 0$ such that
$$|s(x,t,t;\theta_s)| \leq K_s(1+|x|), \quad \text{for all } t \geq 0, x \in \mathbb{R}^{d_1}.$$
When `tanh` function is applied at the last layer,
$$|s(x,t,t;\theta_s)| \leq K_s, \quad \text{for all } t \geq 0, x \in \mathbb{R}^{d_1}.$$
- For any neural network $s(x, t, t; \theta_s) : \mathbb{R}^{d_1} \times \mathbb{R}_+ \times \mathbb{R}_+ \to \mathbb{R}^{d_2}$ with parameter θ_s, there exist positive constants $m, b > 0$ such that for all $t \geq 0$,
$$\langle s(x,t,t;\theta_s), x\rangle \leq -m|x|^2 + b.$$

Above assumptions imply that neural networks of drift function f satisfy the linear growth condition. Similarly, the assumptions of diffusion function can also be obtained. Furthermore, the solution $Z_k(t)$ of Eq. (8) exists uniquely. This can apply to all other events. Therefore, the process $\{\mathbf{Z}(t)\}_{t\in[0,t_1]}$ is clearly the unique solution of Eq. (8) in the time interval $[0, t_1]$. Then, at jump time t_1, $Z_k(t_1^+) = Z_k(t_1) + \Delta Z_k(t_1)$ for all event types by the continuity of functions ρ^n, ρ^v, ρ^e in Eq. (14). To consider the time interval $(t_1, t_2]$, we can set a new $\{\bar{Z}_k(0)\}_{k=1}^{V} = \{Z_k(t_1^+)\}_{k=1}^{V}$, $\bar{W}_k = \{\bar{W}_k(t)\}_{t\geq 0}, \bar{W}_k(t) = W_k(t_1 + t) - W_k(t)$, and $\bar{\mathbf{N}} = \{\mathbf{N}(t)\}_{t\geq 0}, \bar{\mathbf{N}}(t) = \mathbf{N}(t_1 + t) - \mathbf{N}(t)$. Then, on the time interval $(t_1, t_2]$, $\{\mathbf{Z}(t)\}_{t\in(t_1,t_2]} = \{\bar{\mathbf{Z}}(t - t_1)\}_{t\in(t_1,t_2]}$, the solution $\bar{\mathbf{Z}}(t)$ also exists uniquely, which can be proved in the same way as that on the time interval $[0, t_1]$.

By repeating the process, it is eventually possible to determine $\mathbf{Z}(t)$ for t in the interval $[0, t_i]$ for every i, which means a unique adapted left continuous process $\{\mathbf{Z}(t)\}_{t\geq 0}$ with right-limits that satisfies Eq. (15). Moreover, since $\eta : \mathbb{R}^{d_Z} \to \mathbb{R}$ and `Softplus` are also Lipschitz continuous, the resulting process $\{\boldsymbol{\lambda}(t)\}_{t\geq 0}$ also has a unique strong solution, where $\boldsymbol{\lambda}(t) = (\lambda_1(t), \dots, \lambda_V(t))^{\mathrm{T}}$, $\lambda_k(t) = \texttt{Softplus}(\eta(Z_k(t)))$. □

B Description of Datasets

We evaluate our proposed approaches and other baselines on five real-world benchmark datasets used in SOTA methods VAETPP [Yang and Zha, 2024] and NJDTPP [Zhang *et al.*, 2024], including

- New York Motor Vehicle Collisions (NYMVC): This dataset comprises a compilation of vehicle collision events transpiring in New York City since April 2014. Particularly, a vehicle collision might lead to a series of subsequent collisions within the same or nearby districts in a short span of time.
- MathOF and AskUbuntu: Two stack exchange datasets from distinct sources are incorporated in the experiments. These datasets entail diverse interactions among the participants. Such interactions among users typically demonstrate a degree of clustering effects and periodic trends. For example, queries related to trending technology topics may promptly elicit numerous responses or comments from others sharing similar interests.

- Taobao: This public dataset was created and released for the 2018 Tianchi BigData Competition. It contains time-stamped behavior records (e.g., browsing, purchasing) of anonymized users on the online shopping platform Taobao. Each event type represents a user purchasing different types of products.
- Taxi: This dataset includes time-stamped taxi pick-up and drop-off events across the five boroughs of New York City. Each combination of boroughs, whether it involves a pick-up or drop-off, specifies an event type, resulting in a total of 10 event categories.

C Description of Baselines

We compare our approaches with the following eight state-of-the-art models. Meanwhile, in the descriptions of NJSDE and NJDTPP, we emphasize the differences between our method and them. The specific descriptions are as follows.

- **RMTPP** [1] [Du *et al.*, 2016]: RMTPP models the intensity function of a temporal point process as a nonlinear function of the history and uses a recurrent neural network to automatically learn a representation of influences from the event history.
- **FullyNN** [2] [Omi *et al.*, 2019]: FullyNN utilizes a feed-forward neural network to model the integral of the intensity function, subsequently deriving the intensity function, which can precisely evaluate the log-likelihood function and incorporate the intensity function's integral without resorting to numerical approximations.
- **NJSDE** [3] [Jia and Benson, 2019]: NJSDE provides a data-driven approach to learn hybrid systems that involve both flow and jump dynamics. The neural ODEs framework is extended with discontinuities to model TPPs. As mentioned in the related work, our approach differs from NJSDE in two significant ways. Firstly, we employ multidimensional SDEs to model the hidden states of each event type, whereas NJSDE uses ODEs to model the whole system. Secondly, NJSDE does not incorporate the underlying relationships and influences of event types in its jump term. Instead, it directly models the impact of event occurrences on the entire system. Our proposed method enhances the correlations between event types by introducing a network inference module, reducing unnecessary influences between unrelated events, thereby improving model interpretability and predictive capability.
- **THP** [4] [Zuo *et al.*, 2020]: THP utilizes the self-attention mechanism to capture extended dependencies efficiently and models the intensity function using a Transformer architecture.
- **AttNHP** [5] [Yang *et al.*, 2022]: AttNHP extends the Transformer architecture to model sequences of events. The framework constructs comprehensive embeddings of both observed and potential events at any time point, leveraging detailed representations of these events and their contextual information.
- **ITHP** [6] [Meng *et al.*, 2024]: ITHP inherits the advantages of THP while aligning with statistical nonlinear Hawkes processes, improving interpretability and offering valuable insights into user or group interactions. Furthermore, ITHP enhances the adaptability of the intensity function across non-event intervals, enhancing its ability to capture intricate event propagation patterns within social networks.
- **VAETPP** [7] [Yang and Zha, 2024]: VAETPPs utilizes a novel variational auto-encoder to capture a mixture of temporal dynamics. The event dynamics are assumed to be stationary within each sub-interval but could vary across these sub-intervals. The latent dependency graph is learned for each sub-interval to eliminate the non-contributory influences of past events.
- **NJDTPP** [8] [Zhang *et al.*, 2024]: NJDTPP employs a neural jump-diffusion SDE in which the drift, diffusion, and jump coefficient functions are defined by neural networks. This approach showcases model adaptability in capturing intensity dynamics without being constrained by predetermined functional forms. As mentioned in the related work, our approach has two improvements over NJDTPP. Firstly, we employ multi-dimensional SDEs to model the internal features of each event, while enhancing the capture of complex temporal patterns by incorporating past event occurrence times into the drift term. Secondly, we introduce latent graph learning in the jump term and design three additional functions to further enhance the interaction effects between event types.

D Implementation Details

D.1 Training Algorithm

The pseudo-codes of training algorithm of our proposed approach with Eq. (8) as shown in Algorithm 1.

D.2 Training Details

We initialize each initial value z_k of every hidden state and the parameters in the MLPs using a Gaussian distribution. The initial value of Ψ is set to 0. During the training process, we limit the gradient norm of iterable parameters to a value of 1. Additionally, we implement early stopping, which entails terminating training early if the performance on the validation set does not improve for M epochs. In our experiments, M is set to 20. The activation function chosen for these networks is `Tanh`, and we utilize the Adam optimizer with a weight decay of 10^{-5}. In configuring the network parameters for our approach, we do not use hyper-parameter tuning methods such as grid search. Consequently, we believe that better results can be achieved through additional careful tuning.

[1]https://github.com/Hongrui24/RMTPP-pytorch

[2]https://github.com/omitakahiro/NeuralNetworkPointProcess

[3]https://github.com/000Justin000/torchdiffeq/tree/jj585

[4]https://github.com/SimiaoZuo/Transformer-Hawkes-Process

[5]https://github.com/yangalan123/anhp-andtt

[6]https://github.com/waystogetthere/Interpretable-Transformer-Hawkes-Process

[7]https://github.com/sikunyang/VAETPP

[8]https://github.com/Zh-Shuai/NJDTPP

Algorithm 1 Trainning of our approach

```
Input: Model parameter $t_0 = 0$, $L$ multi-type event sequences $\{\mathcal{S}^l\}_{l=1}^L$, $\mathcal{S}^l = \{(v_i, t_i)\}_{i=1}^{n_l}$.
Parameter: $\theta = \{\boldsymbol{\theta}_f, \boldsymbol{\theta}_g, \boldsymbol{\theta}_\eta, \boldsymbol{\theta}_{\rho^n}, \boldsymbol{\theta}_{\rho^e}, \boldsymbol{\theta}_{\rho^v}, \mathbf{z}, \boldsymbol{\Psi}\}$
 1: Initialize $\theta = \{\boldsymbol{\theta}_f, \boldsymbol{\theta}_g, \boldsymbol{\theta}_\eta, \boldsymbol{\theta}_{\rho^n}, \boldsymbol{\theta}_{\rho^e}, \boldsymbol{\theta}_{\rho^v}, \mathbf{z}$ from a Gaussian distribution, $\Psi = \mathbf{0}$, negative log-likelihood $\mathcal{L} = 0$, $Z_k(t_0^+) = z_k$ for each event type $k$.
 2: while not converge do
 3:   for each sequence $\mathcal{S}^l$ in batch do
 4:     for $i = 1, \ldots, n_l + 1$ do
 5:       $\{\mathbf{Z}(\tau_d^i)\}_{d=1}^D$ = SDESolve($f, g, \mathbf{Z}(t_{i-1}^l), (\tau_0^i = t_{i-1}^l, \ldots, \tau_D^i = t_{i-1}^l)$) by using Euler-Maruyama scheme as Eq. (16).
 6:       $\{\boldsymbol{\lambda}(\tau_d^i)\}_{d=1}^D = \texttt{Softplus}(\eta(\{\mathbf{Z}(\tau_d^i)\}_{d=1}^D))$.
 7:       $\mathbf{Z}(t_i^{l+}) = \mathbf{Z}(t_i^l) + \Delta\mathbf{Z}(t_i^l)$ by Eq. (14).
 8:     end for
 9:     $\mathcal{L}^l = -l(\{\boldsymbol{\lambda}(\tau_d^i)\}_{k=0,i=1}^{D,n_l+1})$ by Eq. (17).
10:     $\mathcal{L} += \mathcal{L}^l$.
11:   end for
12:   Back-propagate with gradient $\nabla_\theta \mathcal{L}$.
13:   Clip gradient by norm.
14:   Update model parameters $\theta$ by Adam optimizer.
15: end while
16: return solution
```

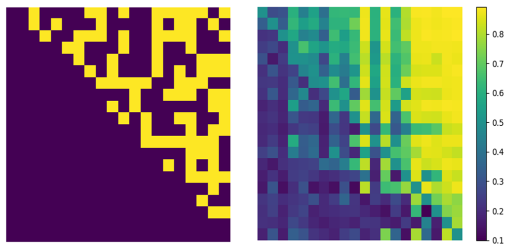
In addition, our experiments are conducted on a Linux server equipped with an A6000 GPU. All code is primarily implemented using the Python language and the PyTorch library. Our methods are run 10 times independently, with the average results shown in Tables 2 and 3. Regarding standard deviation, further clarification is provided here. For the evaluation of NLL, the standard deviation of neural SDE is below 0.03, while that of neural LNSDE is below 0.02. In the evaluation of RMSE, the standard deviations of neural SDE and neural LNSDE are both below 0.003. For the evaluation of Accuracy, the standard deviations of neural SDE and neural LNSDE are both below 0.01. In the evaluation of F_1 scores, the standard deviations of neural SDE and neural LNSDE are both below 0.015.

E Latent Network Inference and Model Interpretability

The synthetic network used is a directed asymmetric network. The original adjacency graph and the inferred adjacency based on our method's latent graph inference module are shown in the Fig. 1. It is quite apparent that our proposed method has to some extent captured the underlying relationships between events.

Meanwhile, we showcase the graph structure learned from the real-world Taobao dataset. Fig. 2 illustrates the degree of connections between different categories of products, highlighting the asymmetric and relatively sparse of these relationships.

Additionally, it is important to emphasize that the inferred relationships between events do not refer to the ground-truth causal relationships but rather to Granger causality, which aims to evaluate the effectiveness of one variable in predicting other variables. In reality, inferring large-scale networks is extremely challenging. For example, as mentioned in [Prasse and Van Mieghem, 2022], there are numerous surrogate adjacencies that can be solved for the low-dimensional dynamics of the nodal state that can provide accurate predictions. This also means that even in cases where the predicted results are good, the estimated adjacency graph may still differ from the ground truth. Therefore, our ultimate goal is to leverage the exploration of latent relationships to enhance modeling and predictive capabilities rather than accurately inferring the ground-truth network structures. The results in the experimental section demonstrate the high accuracy and strong competitiveness of our method in predicting events and times compared to existing state-of-the-art methods.



(a) The original adjacency (b) The inferred adjacency

Figure 1: The original adjacency graph and the inferred adjacency.

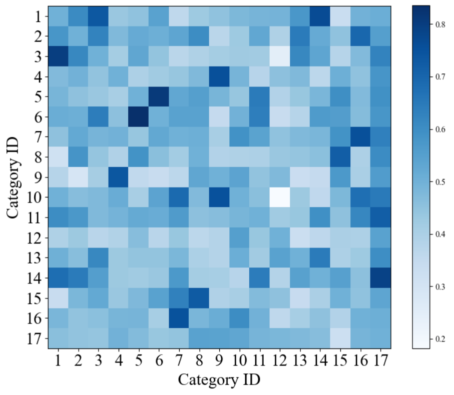


Figure 2: The estimated adjacency on Taobao dataset.

References

[Du *et al.*, 2016] Nan Du, Hanjun Dai, Rakshit Trivedi, Utkarsh Upadhyay, Manuel Gomez-Rodriguez, and Le Song. Recurrent marked temporal point processes: Embedding event history to vector. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1555–1564, 2016.

[Jia and Benson, 2019] Junteng Jia and Austin R Benson. Neural jump stochastic differential equations. *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, pages 9847–9858, 2019.

[Meng *et al.*, 2024] Zizhuo Meng, Ke Wan, Yadong Huang, Zhidong Li, Yang Wang, and Feng Zhou. Interpretable transformer hawkes processes: Unveiling complex interactions in social networks. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 2200–2211, 2024.

[Oh *et al.*, 2024] YongKyung Oh, Dongyoung Lim, and Sungil Kim. Stable neural stochastic differential equations in analyzing irregular time series data. In *Proceedings of the 12th International Conference on Learning Representations*, 2024.

[Omi *et al.*, 2019] Takahiro Omi, Kazuyuki Aihara, et al. Fully neural network based model for general temporal point processes. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, volume 32, pages 2122 – 2132, 2019.

[Prasse and Van Mieghem, 2022] Bastian Prasse and Piet Van Mieghem. Predicting network dynamics without requiring the knowledge of the interaction graph. *Proceedings of the National Academy of Sciences*, 119(44):e2205517119, 2022.

[Yang and Zha, 2024] Sikun Yang and Hongyuan Zha. A variational autoencoder for neural temporal point processes with dynamic latent graphs. In *Proceedings of the 38th AAAI Conference on Artificial Intelligence*, volume 38, pages 16343–16351, 2024.

[Yang *et al.*, 2022] Chenghao Yang, Hongyuan Mei, and Jason Eisner. Transformer embeddings of irregularly spaced events and their participants. In *Proceedings of the 10th International Conference on Learning Representations*, 2022.

[Zhang *et al.*, 2024] Shuai Zhang, Chuan Zhou, Yang Aron Liu, Peng Zhang, Xixun Lin, and Zhi-Ming Ma. Neural jump-diffusion temporal point processes. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pages 60541–60557. PMLR, 2024.

[Zuo *et al.*, 2020] Simiao Zuo, Haoming Jiang, Zichong Li, Tuo Zhao, and Hongyuan Zha. Transformer hawkes process. In *Proceedings of the 37th International Conference on Machine Learning*, pages 11692–11702. PMLR, 2020.