ISSN 2616-5775 Vol. 7, Issue 11: 20-26, DOI: 10.25236/AJCIS.2024.071103

# A Retentive Hawkes Process for Long Event Sequence Prediction

**Zeyu Huang[1,a], Zhaoman Zhong[1,b,*], Xinru Cui[1,c]**

*[1]School of Computer Engineering, Jiangsu Ocean University, Lianyungang, China*
*[a]2022210905@jou.edu.cn, [b]zhongzhaoman@163.com, [c]2022210903@jou.edu.cn*
*[*]Corresponding author*

***Abstract:*** *Predicting event sequences is crucial across various domains. However, most existing transformer-based point process models struggle with longer sequences due to their quadratic memory complexity. To address this, we propose the Retentive Hawkes Process (RHP) model. The RHP uses a retention mechanism to simplify computations, enable a recurrent formulation, resulting in linear memory complexity and reduced inference latency while effectively modeling the self-exciting nature of event sequences and capturing both temporal dynamics and long-range dependencies. Numerical experiments demonstrate that RHP significantly outperforms traditional Transformer-based models and Hawkes Processes variants across diverse datasets. Furthermore, RHP shows promising scaling results in computational paradigms.*

***Keywords:*** *hawkes process; event prediction; retention mechanism*

## 1. Introduction

Analyzing event sequences is crucial for understanding complex temporal dynamics in finance, healthcare, and social networks. The Hawkes process is notable for capturing self-exciting behavior, where an event increases the likelihood of subsequent events in a short time. However, traditional Hawkes processes struggle with long-term dependencies and typically assume memoryless excitation, where past events lose influence beyond a certain time frame.

While RNN-based models have made strides in likelihood estimation and event sequence prediction, they face challenges in capturing long-term dependencies and suffer from poor trainability. Transformer-based models, utilizing self-attention mechanisms, effectively capture both short- and long-term dependencies and provide computational efficiency. However, they encounter inefficient inference and difficulties with very long sequences due to their quadratic complexity.

To address these issues, we propose the Retentive Hawkes Process (RHP), which integrates a retention mechanism with the classic Hawkes process. This mechanism allows the model to retain memory of past events over extended periods, improving its ability to capture long-range dependencies and complex interactions between events. RHP is particularly valuable for scenarios where events are influenced not only by recent occurrences but also by longer sequences. Additionally, it supports parallel training and offers low-cost inference. The main contributions of our work are as follows:

1) We introduce a novel recurrent model that integrates attention mechanisms, recurrent structures, and Hawkes Process techniques to effectively address this problem.

2) To the best of our knowledge, the RHP model is the first to utilize the output of a retention mechanism to redefine the conditional intensity function for predicting future events.

3) The results across datasets of varying lengths demonstrate that the proposed model successfully captures short-term, long-term, and even extended dependencies.

4) To validate the efficiency of our model, we employ three distinct metrics: log-likelihood, accuracy, and root-mean-square error. The numerical experiments show that the proposed model outperforms other Hawkes Process models.

## 2. Preliminaries

We will briefly review Hawkes Process[1], Transformer Hawkes Process[2], and Retentive Network[3] in this section.

**Hawkes Process**, introduced by Alan G. Hawkes in 1971, is defined by its intensity function, which captures the rate at which events occur over time. Mathematically, the intensity function $\lambda(t)$ of a Hawkes Process can be represented as:

$$\lambda(t) = \mu + \sum_{t_i<t} \phi(t - t_i) \tag{1}$$

**Transformer Hawkes Process** incorporates the temporal modeling capabilities of the Hawkes Process with the representational power and flexibility of Transformers:

$$p(t|\mathcal{H}_t) = \lambda(t|\mathcal{H}_t) exp\left(-\int_{t_j}^{t} \lambda(\tau|\mathcal{H}_\tau)d\tau\right) \tag{2}$$

$$\widehat{t_{j+1}} = \int_{t_j}^{\infty} t \cdot p(t|\mathcal{H}_t)dt \tag{3}$$

$$\widehat{k_{j+1}} = \underset{k}{\operatorname{argmax}} \frac{\lambda_k(t_{j+1}|\mathcal{H}_{j+1})}{\lambda(t_{j+1}|\mathcal{H}_{j+1})} \tag{4}$$

The prediction targets are $\widehat{t_{j+1}}$ and$\widehat{\mathrm{k}_{\mathrm{j}+1}}$. Here, $\lambda(t)$ is the intensity function, K is the number of event types, and $\mathcal{H}_t$ are the hidden states of the event sequence, obtained through a Transformer module. While the Transformer architecture has excelled in natural language processing, enabling continuous-time outputs, it struggles with training recurrently and balancing the capture of long-term dependencies with efficiency, which are limitations of the Transformer Hawkes Process.

**Retentive Network** offers a novel solution, enhancing neural networks' ability to retain and utilize past information. RetNet's retention mechanisms help address the limitations of Transformers. However, this architecture requires adaptation for point process modeling, as event sequences have irregular time intervals, unlike the regular spacing of words in natural language. Thus, RetNet must be generalized to operate in a continuous-time domain.

## 3. Retentive Hawkes Process

We introduce our proposed Retentive Hawkes Process. Given a specific event $s_j = (t_j, k_j)$ where $\mathrm{k_j}$ for type and $t_j$ for time, let $\mathcal{S} = \{\mathrm{s_j}\}_{\mathrm{j}=1}^{\mathrm{L}} = \{(\mathrm{t_j}, \mathrm{k_j})\}_{\mathrm{j}=1}^{\mathrm{L}}$ be a realization of an event sequence of L events, the input vectors $\mathcal{S}$ is first packed into $S^0 = [s_1, \cdots, s_{|L|}] \in R^{|L| \times d_{model}}$, where $d_{model}$ is hidden dimension, then we compute event vector representations $S^l = \mathrm{RetNet_l}(S^{l-1}), l \in [1, L]$. Figure 1 illustrates the architecture of RHP.

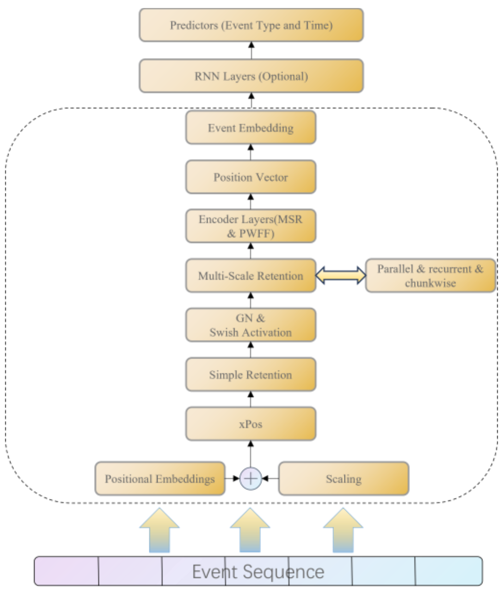

*Figure 1: Architecture of the Retentive Hawkes Process.*

We apply the retentive network to the domain of event sequences, incorporating recurrent structures while preserving self-attention mechanisms to ensure computational efficiency by integrating time series characteristics. Formally, the input sequence $\mathcal{S}$ is transformed into vectors by a word embedding layer. We then use the packed embeddings $\mathcal{S}^0$ as input and compute the model output $\mathcal{S}^L$:

$$Y^l = \mathrm{MSR}\left(\mathrm{LN}(\mathcal{S}^\ell)\right) + \mathcal{S}^\ell \tag{5}$$

$$\mathcal{S}^{\ell+1} = \mathrm{FFN}\left(\mathrm{LN}(Y^l)\right) + Y^l \tag{6}$$

In encoder layer (dotted box in Figure 1, our model has two different encoding procedure for temporal information Trigonometric functions[2] and xPos encoding[4]. Trigonometric functions is primarily used to capture the temporal position of events within a sequence using a fixed sinusoidal positional encoding scheme.

$$\mathrm{Tri}(t_j) = \sin\left(\frac{t_j}{10000^{2i/d}}\right), \cos\left(\frac{t_j}{10000^{2i/d}}\right) \tag{7}$$

Let $t_j$ be the input vector, where j denotes the position, and $d$ represents the dimension of each head in the multi-head attention mechanism. α is the scaling factor. Define $k_j$ as the one-hot encoding[2]. Let $K = [k_1, k_2, \dots, k_L]$ and $Z = [Tri(t_1), Tri(t_2), \dots, Tri(t_L)] \in R^{M\times L}$. For any event and its corresponding timestamp $(t_j, k_j)$, the temporal encoding is represented by $Z$, and the event embedding is given by $UK$, where $U$ is an embedding matrix. The embedding of the event sequence $\mathcal{S} = (t_j, k_j)_{j=1}^{L}$ is then defined as follows:

$$S = (UK + Z)^\top \tag{8}$$

xPos Encoding modifies the query $Q$ and key $K$ vectors to incorporate positional information:

$$xPos(s) = (s \cdot \cos(\theta_s)) + (\mathrm{rotate}(s) \cdot \sin(\theta_s)), \text{ where } \theta_s = \frac{(j + 0.4d)}{1.4d \cdot \alpha} \tag{9}$$

$$Q = xPos(S \cdot W_Q) \tag{10}$$

$$K = xPos(S \cdot W_K) \tag{11}$$

$$V = S \cdot W_V \tag{12}$$

Respectively, $W_Q$, $W_K$ and $W_V$ are different weight matrices for the query, key and value vectors.The xPos encoding modifies these vectors to include positional information. After applying xPos encoding, the modified $Q$, $K$, $V$ matrix are used in the retention mechanism. The retention mechanism calculates an attention score using the dot product of $W_Q$ and $W_K$, which is then scaled by a positional scaling factor $D$ and $V$, the final output of the retention mechanism is:

$$\mathrm{Retention}^l(Q, K, V) = (Q \cdot K^T) \cdot D \cdot V, \text{where } D = \gamma^{|i-j|} \tag{13}$$

Here, $D$ is the positional scaling factor matrix with γ representing the decay factor which is same as RetNet[3], and $V$ is the value vector obtained by applying another weight matrix $W_V$ to the input $S$.

$$d_i = \sum_{i=1}^{\mathrm{d}} \mathrm{Retention}(Q_i, K_i, V_i, D_i) \tag{14}$$

$$Y = \mathrm{GroupNorm}\left(\mathrm{reshape}\left(\mathrm{Concat}(d_i)\right)\right) \tag{15}$$

$$MSR(S) = (\mathrm{swish}(S \cdot W_G) \cdot \mathrm{Y}) \cdot W_O \tag{16}$$

Here, $W_G$ and $W_O$ are learnable, GroupNorm[5] normalizes the output of each head. These outputs are concatenated along the feature dimension to form a combined output $Y$. A swish gate[6][7] is added to increase non-linearity in the retention layers. Each head uses multiple γ scales, resulting in distinct variance statistics, which require separate normalization for each head's output. Figure 2 illustrates the architecture of Retentive Network.

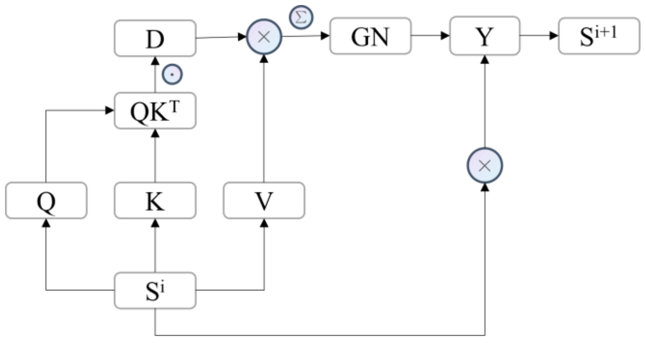

*Figure 2: Architecture of the Retentive Network.*

Each head in the MSR mechanism computes a retention score matrix, capturing attention weights across the sequence. This output is then fed into layers like the Feed-Forward Network (FFN) for further processing. A Layer Normalization (LN) step adds non-linearity and higher-order interactions.

We use the softplus activation function to capture nuanced patterns in the data. The final output, obtained after the FFN, provides a rich representation of the input sequence, capturing local and global dependencies across temporal scales. The hidden representations of all events are denoted by $H = FFN(S)$, where each row corresponds to a specific event.

In time series forecasting, each head learns exclusively from past events through three mechanisms: causal masking prevents future information leakage; the retention mechanism decays a matrix $D$ based on temporal distance; sequential processing iteratively updates the state $S$ or $Y$ using only prior information.

From eq.(7) to eq.(16), temporal dependencies in the sequence are captured via the retention mechanism, generating hidden representations $H$ for the Hawkes process.

## 4. Experiments

We compare the performance of the RHP against several existing models: the Recurrent Marked Temporal Point Process[8], Neural Hawkes Process[9], Time Series Event Sequence[10], Self-Attentive Hawkes Processes[11], Transformer Hawkes Process[2], Hyperbolic Geometric Transformer Hawkes Process[12]. The models are evaluated based on per-event log-likelihood and event prediction accuracy on test sets. We evaluate the model on the Nvidia 4090D-24GB GPU in our experiments.

### *4.1. Datasets*

We evaluate the models using several real-world datasets: Retweets[13]Sequences of tweets. MemeTrack[14]: The life cycle of a specific meme. Financial Transactions[8]: Transaction records for a single stock over the course of one day. Electrical Medical Records[15]: The MIMIC-II medical dataset records patient visits to a hospital's ICU over a seven-year period. Stack Overflow[14]: A question-and-answer website.

### *4.2. Competing methods*

To evaluate the predictive performance of our forecasting methods, we compare the Recurrent Hawkes Process (RHP) with several Hawkes process-based models. Specifically, we examine the Temporal Hawkes Process (THP) in Section 2, along with the other models detailed below:

**RMTPP**[8]: It provides a strong framework for modeling and predicting time-based events with specific attributes. Its ability to handle event recurrence and markers makes it useful for analyzing complex temporal patterns and forecasts.

**NHP**[9]: By integrating neural networks, it improves the traditional Hawkes process to model non-linear temporal dependencies. This significantly enhances forecasting accuracy in dynamic systems.

**TSES**[10]: It captures temporal dependencies in event sequences using recurrent architectures. This method improves the accuracy and flexibility of temporal pattern analysis and forecasting.

**SAHP**[11]: Self-attention mechanisms enhance the traditional Hawkes process, improving understanding of event interactions and temporal dependencies. This leads to better predictions of long-range and complex patterns in dynamic environments.

**HGTHP**[12]: Combining hyperbolic geometry with transformers, this model represents complex temporal and relational structures more effectively. This results in higher prediction accuracy and deeper insights into event dynamics.

### *4.3. Training Details and Evaluation Metrics*

We selected the RMSE to assess the accuracy of time predictions made by the RHP, and used log-likelihood and accuracy as the metric for evaluating event marker predictions, the Adam[16] served as our optimizer, while StepLR[17] was employed as the learning rate scheduler.

### *4.4. Likelihood Comparison*

We compared the proposed model to six existing methods using Log-likelihood, accuracy, and RMSE metrics on real-world datasets. Log-likelihood results are in Figure 3 and Table 1, accuracy and RMSE in Table 2. Note that likelihood-free models like TSES are excluded from Log-likelihood comparisons.

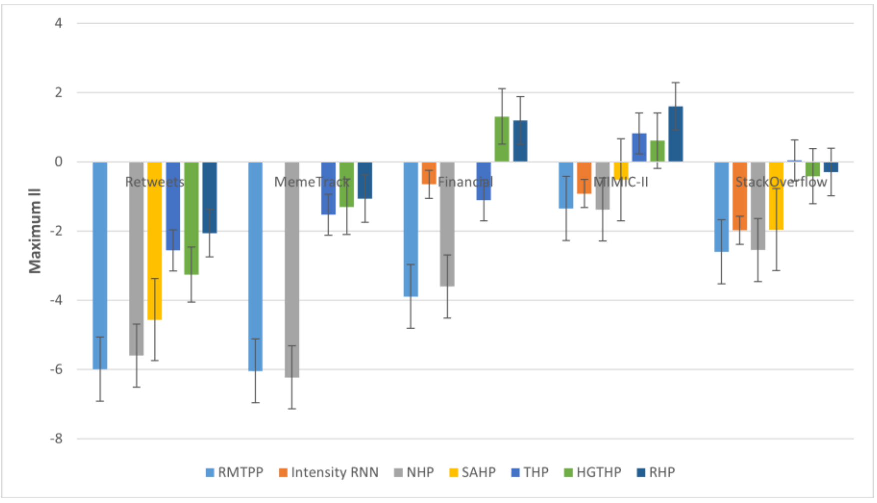

*Figure 3: Architecture of the Retentive Network.*

*Table 1: Log-likelihood comparison.*

| Model | Retweets | Meme | Financial | MIMIC-II | SO |
|---|---|---|---|---|---|
| RMTPP | -5.99 | -6.04 | -3.89 | -1.35 | -2.6 |
| Intensity_RNN | - | - | -0.65 | -0.92 | -1.98 |
| NHP | -5.6 | -6.23 | -3.6 | -1.38 | -2.55 |
| SAHPP | -4.56 | - | - | -0.52 | -1.96 |
| THP | -2.26 | -1.53 | -1.11 | 0.82 | 0.042 |
| HGTHP | -3.26 | -1.3 | 1.31 | 0.61 | -0.42 |
| **RHP** | **-2.06** | **-1.06** | **1.2** | **1.6** | **-0.3** |

Figure 3 and Table 1 show performance across six datasets. Our model consistently achieves higher Log-likelihood than most baselines.

In the Retweets dataset, with long sequences (average length 109), RHP slightly outperforms THP in capturing long-term dependencies. For shorter sequences, like in MemeTrack (average length 3) and MIMIC-II (average length 4), RHP effectively handles short-term dependencies, showing significant improvement in MIMIC-II but performing closer to the baseline in MemeTrack. We attribute this difference to the distinct nature of the datasets: MemeTrack captures the life cycle of a meme, while MIMIC-II involves patient visit records.

In the Financial dataset, RHP's performance is comparable to HGTHP and exceeds THP. In the Stack Overflow dataset (average length 72), differences between models are minimal due to shorter sequences limiting the ability to showcase their strengths. We selected the RMSE to assess the accuracy of time predictions made by the RHP, and used log-likelihood and accuracy as the metric for evaluating event marker predictions, the Adam[16] served as our optimizer, while StepLR[17] was employed as the learning rate scheduler.

We present two datasets as examples to demonstrate the model's ability to handle longer sequences. To ensure generalization, we did not select the best-performing training run but instead chose a random training instance from the MemeTrack dataset and the Financial dataset, which have average sequence lengths of 3 and 2074, respectively. As shown in Figure 4.

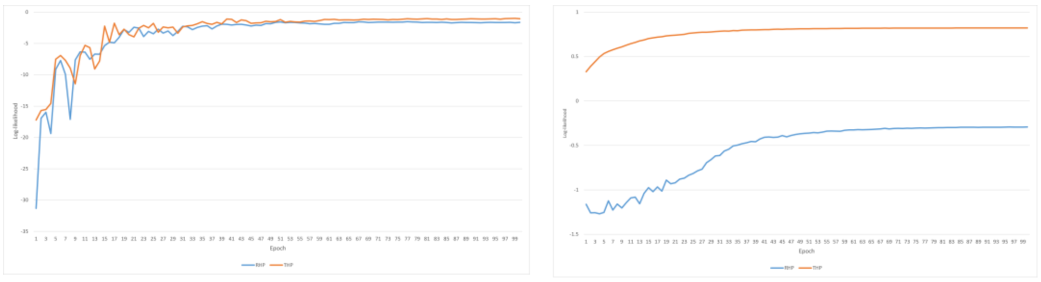

*Figure 4: Log-likelihood comparison of THP(blue) and RHP(orange) on MemeTrack and Financial dataset.*

In the MemeTrack dataset, due to the short sequence length, the performance gap between the two models is not significant. However, RHP converges faster than THP, achieving superior final results with smaller fluctuations in log-likelihood. In contrast, in the Financial dataset, where sequences are much longer, there is a substantial performance gap between the two models. RHP not only outperforms THP in convergence speed but also in the final results, showcasing its effectiveness in optimizing long sequences.

RHP's integration of a recurrent mechanism with the Transformer architecture enables it to scale more efficiently with sequence length, better maintain and propagate information over time, and reduce computational and memory demands. These advantages make RHP a more effective model than traditional Transformer-based approaches for tasks involving long sequences.

### *4.5. Event Prediction Comparison*

According to eq.(3) and eq.(4), in addition to Log-likelihood, we predict $\widehat{t_{j+1}}$ and $\widehat{k_{j+1}}$ for every event $s_j = (t_j, k_j)$. Event type predictions are evaluated using accuracy, and event time predictions with Root Mean Square Error (RMSE). The results are summarized in Table 2.

*Table 2: Event type prediction accuracy (left) and Event time prediction RMSE(right).*

| Model | FIN | MIMIC-II | SO |
|---|---|---|---|
| RMTPP | 61.95 | 81.2 | 45.9 |
| NHP | 62.2 | 83.0 | 46.3 |
| TSES | 62.17 | 83.0 | 46.2 |
| THP | 62.64 | 85.3 | 47.0 |
| HGTMP | 61.9 | 84.9 | 46.9 |
| **RHP** | **62.7** | **85.3** | **47** |

| Model | FIN | MIMIC-II | SO |
|---|---|---|---|
| RMTPP | 1.56 | 6.12 | 9.78 |
| NHP | 1.56 | 6.13 | 9.83 |
| TSES | 1.5 | 4.7 | 8.0 |
| SAHP | - | 3.89 | 5.57 |
| THP | 0.93 | 0.82 | 4.99 |
| HGTMP | 0.41 | 0.74 | 4.01 |
| **RHP** | **0.5** | **0.7** | **3.89** |

RHP consistently outperforms the baselines across tasks. In the financial dataset, it surpasses most models in several metrics but performs worse than HGTHP in RMSE. This is because Log-likelihood reflects how well the model captures the data distribution, while RMSE focuses on prediction accuracy. RHP may fit the overall distribution well (high Log-likelihood) but produce less accurate individual predictions (higher RMSE), possibly due to overconfidence or sensitivity to outliers. In the MIMIC-II and Stack Overflow datasets, RHP slightly outperforms other models in both accuracy and RMSE.

Overall, the results demonstrate that RHP effectively captures short-term, long-term, and extended dependencies better than existing methods. We selected the RMSE to assess the accuracy of time predictions made by the RHP, and used log-likelihood and accuracy as the metric for evaluating event marker predictions, the Adam[16] served as our optimizer, while StepLR[17] was employed as the learning rate scheduler.

## 5. Conclusions

In this paper we present Retentive Hawkes Process, by addressing the challenges identified in existing work, RHP offers a powerful and efficient solution for analyzing complex event sequences, providing enhanced fidelity and insight into temporal dynamics across a wide range of applications. The empirical studies presented in this paper demonstrate the superior performance of RHP on real-world datasets, underscoring its potential as a robust tool for long event sequence temporal modeling.

## Acknowledgements

The authors acknowledge the support from the National Natural Science Foundation of China (72174079), Jiangsu Province "Qinglan Project" Excellent Teaching Team for Big Data (2022-29) and Lianyungang City Key Research and Development Plan (Industry Foresight and Key Core Technologies) Project(CG2323).

## References

*[1] Hawkes A G. Spectra of some self-exciting and mutually exciting point processes[J]. Biometrika, 1971, 58(1): 83.*
*[2] Zuo S, Jiang H, Li Z, et al. Transformer hawkes process[C]//International conference on machine learning. PMLR, 2020: 11692.*
*[3] Sun, Y.; Dong, L.; Huang, S.; Ma, S.; Xia, Y.; Xue, J.; Wang, J.; Wei, F. Retentive network: A Successor to Transformer for Large Language Models. arXiv 2023, arXiv:2307.08621.*
*[4] Sun Y, Dong L, Patra B, et al. A Length-Extrapolatable Transformer[C]//The 61st Annual Meeting Of The Association For Computational Linguistics. 2023.*
*[5] Wu Y, He K. Group normalization[C]//Proceedings of the European conference on computer vision (ECCV). 2018: 3.*
*[6] Hendrycks D, Gimpel K. Gaussian error linear units (gelus)[J]. arXiv preprint arXiv:1606.08415, 2016.*
*[7] Ramachandran P, Zoph B, Le Q V. Searching for activation functions[J]. arXiv preprint arXiv:1710.05941, 2017.*
*[8] Du N, Dai H, Trivedi R, et al. Recurrent marked temporal point processes: Embedding event history to vector[C]//Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining. 2016: 1555.*
*[9] Mei H, Eisner J M. The neural hawkes process: A neurally self-modulating multivariate point process[J]. Advances in neural information processing systems, 2017, 30:6754.*
*[10] Xiao S, Yan J, Yang X, et al. Modeling the intensity function of point process via recurrent neural networks[C]//Proceedings of the AAAI conference on artificial intelligence. 2017, 31(1).*
*[11] Zhang Q, Lipani A, Kirnap O, et al. Self-attentive Hawkes process[C]//International conference on machine learning. PMLR, 2020: 11183.*
*[12] Xie Y, Wu J. HGTHP: a novel hyperbolic geometric transformer hawkes process for event prediction [J]. Applied Intelligence, 2024, 54(1): 357.*
*[13] Zhao Q, Erdogdu M A, He H Y, et al. Seismic: A self-exciting point process model for predicting tweet popularity[C]//Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining. 2015: 1513.*
*[14] Leskovec J, Krevl A. SNAP Datasets: Stanford Large Network Dataset Collection [OL]. (2014-06). http://snap.stanford.edu/data.*
*[15] Johnson A E W, Pollard T J, Shen L, et al. MIMIC-III, a freely accessible critical care database[J]. Scientific data, 2016, 3(1): 1.*
*[16] Kingma D P. Adam: A method for stochastic optimization[J]. arXiv preprint arXiv:1412.6980, 2014.*
*[17] Paszke A, Gross S, Massa F, et al. Pytorch: An imperative style, high-performance deep learning library[J]. Advances in neural information processing systems, 2019, 32.*