

# KANformer Hawkes Process

Anuj Kumar

MTech Project Report

## Abstract

Modern data collection processes generate vast quantities of event sequence data across diverse domains such as social media, healthcare, and financial markets. These datasets often exhibit intricate short-term and long-term temporal dependencies. Conventional point process models, relying on recurrent neural networks (RNNs), struggle to accurately capture these dependencies, resulting in unreliable predictions. To address this challenge, we introduce the KANformer Hawkes Process (KHP) model. KHP extends the Transformer Hawkes Process (THP) by replacing the feed-forward layer with Kolmogorov-Arnold Networks (KAN) layers, incorporating spline parameters. This modification harnesses the self-attention mechanism to effectively capture long-term dependencies and accurately model continuous-time event streams. Our KHP model demonstrates superior likelihood and predictive accuracy on both real-world and synthetic datasets, surpassing existing models and exhibiting robustness even in the presence of missing data. Moreover, KHP can integrate additional structural knowledge, enhancing its performance in predicting multiple point processes by leveraging relational information. This represents a significant advancement in modeling the intricate temporal dynamics of continuous-time event streams.

## 1 Introduction

Event sequence data are prevalent in numerous fields such as social media, healthcare, and financial markets, where they generate extensive data with complex temporal dependencies. For instance, social media platforms like Twitter and Facebook produce sequences of events like tweets and comments, while financial transactions and electronic medical records consist of sequences of events at various timestamps. Unlike the time series data, event sequences are usually asynchronous, meaning the time intervals between events play a crucial role in describing their dynamics. Traditional models like the Hawkes process [13] and Poisson point process [14] are often used but rely on simplified assumptions that can limit their practical applicability. For example, the Hawkes process presumes that all past events have a positive influence on the current events, which is not always accurate.

To address the limitations of conventional point process models, advanced methods such as likelihood-free approaches, non-parametric models, and deep neural networks have been proposed. Recurrent Neural Networks (RNNs) [10], along with their variants like LSTM [12] and GRU [11], have shown improvements in modeling event sequence data. However, RNN-based models often struggle with capturing long-

term dependencies and encounter training difficulties due to gradient issues. The Transformer Hawkes Process (THP) [3] model, leverages self-attention mechanism, tackles these challenges by efficiently capturing both long-term and short-term dependencies. The non-recurrent structure of THP allows for efficient training of deeper models, enabling parallel computation and enhanced scalability.

Despite these advancements, there is still room for improvement. Drawing inspiration from the Kolmogorov-Arnold representation theorem [1], we introduce the KANformer Hawkes Process (KHP) model. KHP enhances the THP model by replacing the feed-forward layer with Kolmogorov-Arnold Networks (KAN) layers that utilize spline parameters. KANs offer superior accuracy and interpretability compared to MLPs, addressing the curse of dimensionality by leveraging the compositional structure of functions. Our KHP model demonstrates superior likelihood and predictive accuracy on both real and synthetic datasets, outperforming existing models and exhibiting robustness under missing-data conditions. By incorporating additional structural knowledge, KHP improves prediction performance for learning multiple point processes, representing a significant advancement in modeling complex temporal dynamics in continuous-time event streams.

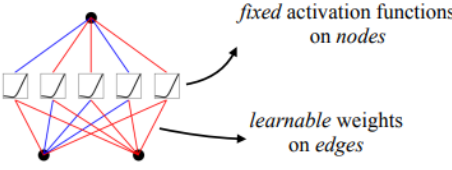
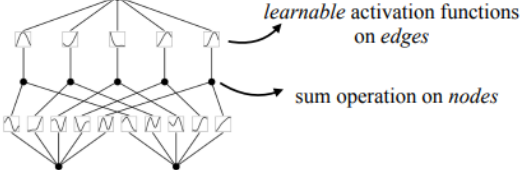
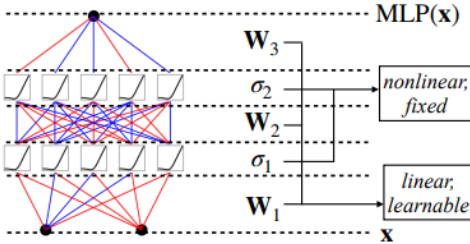
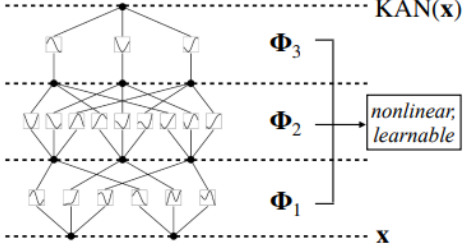
Model	<b>Multi-Layer Perceptron (MLP)</b>	<b>Kolmogorov-Arnold Network (KAN)</b>
Theorem	<b>Universal Approximation Theorem</b>	<b>Kolmogorov-Arnold Representation Theorem</b>
Formula (Shallow)	$f(\mathbf{x}) \approx \sum_{i=1}^{N(e)} a_i \sigma(\mathbf{w}_i \cdot \mathbf{x} + b_i)$	$f(\mathbf{x}) = \sum_{q=1}^{2n+1} \Phi_q \left( \sum_{p=1}^n \phi_{q,p}(x_p) \right)$
Model (Shallow)	(a)  fixed activation functions on nodes learnable weights on edges	(b)  learnable activation functions on edges sum operation on nodes
Formula (Deep)	$\text{MLP}(\mathbf{x}) = (\mathbf{W}_3 \circ \sigma_2 \circ \mathbf{W}_2 \circ \sigma_1 \circ \mathbf{W}_1)(\mathbf{x})$	$\text{KAN}(\mathbf{x}) = (\Phi_3 \circ \Phi_2 \circ \Phi_1)(\mathbf{x})$
Model (Deep)	(c)  MLP(x) $\mathbf{W}_3$ $\sigma_2$ $\mathbf{W}_2$ $\sigma_1$ $\mathbf{W}_1$ $\mathbf{x}$ nonlinear, fixed linear, learnable	(d)  KAN(x) $\Phi_3$ $\Phi_2$ $\Phi_1$ $\mathbf{x}$ nonlinear, learnable

Figure 1: Multi-Layer Perceptrons (MLPs) vs. Kolmogorov-Arnold Networks (KANs)

## 2 Background

In this section, we review the foundational concepts relevant to our work: the Kolmogorov-Arnold Networks [1], the Hawkes Process [13], the Neural Hawkes Process [2], and the Transformer architecture [4]. These form the basis for our proposed model, the Kolmogorov-Arnold Network-based Transformer Hawkes Process (KANformer Hawkes Process).

### 2.1 Kolmogorov-Arnold Networks (KAN)

Drawing inspiration from the Kolmogorov-Arnold representation theorem, we introduce Kolmogorov-Arnold Networks (KANs) as innovative alternatives to Multi-Layer Perceptrons (MLPs), illustrated in figure 1. Unlike MLPs, which utilize the fixed activation functions at the nodes ("neurons"), KANs employ learnable activation functions at the edges ("weights"). In the KANs, linear weights are entirely replaced by univariate functions parameterized as splines, leading to notable improvements in both

accuracy and interpretability.

To formally describe KANs, consider a function  $f : \mathbb{R}^n \rightarrow \mathbb{R}$ . The Kolmogorov-Arnold representation theorem states that every continuous function  $f$  can be represented as:

$$f(x) = \sum_{q=1}^{2n+1} \Phi_q \left( \sum_{p=1}^n \varphi_{q,p}(x_p) \right),$$

where  $\Phi_q$  and  $\varphi_{q,p}$  are continuous functions. In a KAN, the function  $f$  is modeled by learning these univariate functions  $\varphi_{q,p}$  and  $\Phi_q$ , implemented as splines, directly on the network's edges.

In contrast, a conventional MLP with  $L$  layers, each containing  $N$  neurons, approximates  $f$  through a series of linear transformations and fixed nonlinear activations:

$$\text{MLP}(x) = (W_{L-1} \circ \sigma \circ W_{L-2} \circ \sigma \circ \dots \circ W_1 \circ \sigma \circ W_0)x,$$

where  $W_i$  and  $b_i$  are the weights and biases, respectively, and  $\sigma$  is the activation function.

In a KAN network,  $\Phi_l$  represents the function matrix corresponding to the  $l$ th KAN layer. A general KAN network comprises  $L$  layers: for the input vector  $x$ , the output vector of KAN is:

$$\text{KAN}(x) = (\Phi_{L-1} \circ \Phi_{L-2} \circ \cdots \circ \Phi_1 \circ \Phi_0)x.$$

KANs have demonstrated superior accuracy and interpretability compared to MLPs in small-scale AI + Science tasks. For instance, in function fitting tasks, smaller KANs can get comparable or better accuracy than larger MLPs. Both theoretically and empirically, KANs exhibit faster neural scaling laws than MLPs.

The performance of KANs and MLPs has been evaluated on special functions common in mathematics and physics, with KANs consistently showing better Pareto Frontiers, achieving lower training and test losses with fewer parameters compared to MLPs. Some of the special functions are shown in table ??.

In summary, KANs represent an innovative and powerful approach to neural network design, leveraging the Kolmogorov-Arnold representation theorem to enhance both accuracy and interpretability, and demonstrating significant advantages over traditional MLPs in various scientific applications.

## 2.2 Hawkes Process

Introduced by Alan Hawkes in 1971, the Hawkes Process is a self-exciting point process designed to model sequences of discrete events over continuous time. The key characteristic of the Hawkes Process is its intensity function, expressed as:

$$\lambda(t) = \mu + \sum_{j:t_j < t} \psi(t - t_j),$$

where  $\mu$  is base intensity, and  $\psi(\cdot)$  is a decaying function in hawkes process, such as an exponential or power-law function. This formulation implies that each past event positively influences the likelihood of future events, with the influence diminishing over time. While this is suitable for scenarios where events are mutually exciting, a significant limitation is its inability to model inhibitory effects, where past events might reduce the likelihood of future events—a common occurrence in complex real-life situations.

## 2.3 Neural Hawkes Process

To address some of the limitations of classical Hawkes Process, the Neural Hawkes Process [2] incorporates

recurrent neural networks (RNNs) into the model. The intensity function in Neural Hawkes Process is given as:

$$\lambda(t) = \sum_{k=1}^K \lambda_k(t) = \sum_{k=1}^K f_k(w_k^T h(t)),$$

where  $f_k(x) = \beta_k \log(1 + \exp(x/\beta_k))$  is a softplus function ensuring positive intensity values,  $K$  is total number of event types, and  $h(t)$  is hidden states derived from a continuous-time long short term memory (CLSTM) module. The CLSTM adapts the standard LSTM to handle continuous time, facilitating the modeling of event sequences with irregularly spaced event times.

While the Neural Hawkes Process enhances the flexibility of modeling point processes by utilizing the representation power of RNNs, it inherits the intrinsic weaknesses of RNNs, such as difficulty in capturing long-term dependencies and challenges in training due to issues like gradient vanishing and explosion.

## 2.4 Transformer Architecture

The Transformer model, introduced by Vaswani et al. [4], has significantly advanced natural language processing tasks like machine translation and language modeling. It utilizes a self-attention mechanism to capture dependencies within input sequences, facilitating efficient parallel processing and enhanced handling of long-term dependencies.

However, despite its remarkable achievements in natural language processing, the Transformer’s application in other domains, especially in event sequences characterized by irregular time intervals, has been limited. This constraint primarily arises from the standard Transformer’s assumption of regularly spaced inputs, a characteristic not common in most event sequences.

## 2.5 KANformer Hawkes Process

Our proposed model, the KANformer Hawkes Process (KHP), combines the strengths of Kolmogorov-Arnold Networks (KANs) and the Transformer architecture within the framework of the Hawkes Process. KANs leverage the Kolmogorov-Arnold representation theorem, which posits that any multivariate continuous function can be represented as a composition of univariate functions and addition. This enables KANs to use learnable activation functions on edges rather than

Function	Minimal KAN Shape	KAN Test RMSE	MLP Test RMSE
Jacobian Elliptic Functions <code>ellipj(x, y)</code>	[2, 2, 1]	$1.33 \times 10^{-4}$	$6.48 \times 10^{-4}$
Bessel Function of the First Kind <code>jv(x, y)</code>	[2, 2, 1]	$1.64 \times 10^{-3}$	$5.52 \times 10^{-3}$
Spherical Harmonics <code>sph_harm(m, n, x, y)</code>	[2, 1, 1]	$2.21 \times 10^{-7}$	$1.25 \times 10^{-6}$

Performance of KANs and MLPs on Special Functions

fixed activation functions on nodes, enhancing their capability to model complex dependencies in data.

By integrating KANs into the Transformer framework, the KANformer Hawkes Process can adaptively learn dependencies among events over both short and long temporal distances. This is achieved through the self-attention mechanism, which assigns attention scores to pairs of events, indicating the strength of their dependencies. This flexible attention mechanism allows the model to capture intricate patterns in the event sequences, overcoming the limitations of both traditional RNN-based methods and static convolution-based models.

### 3 Model

The KANformer Hawkes Process (KHP) integrates Kolmogorov-Arnold Networks with the Transformer architecture shown in figure 2, specifically within the Hawkes Process framework. Our model captures the temporal dependencies of event sequences while benefiting from the learnable edge activation functions provided by KANs. An event sequence of  $L$  events can be seen as,

$$\mathcal{S} = \{(t_j, k_j)\}_{j=1}^L$$

Here all events has a type  $k_j \in \{1, 2, \dots, K\}$ , representing one of the  $K$  possible types. Thus, each  $(t_j, k_j)$  pair indicates that event of type  $k_j$  occurs at time  $t_j$ .

#### 3.1 Self-Attention with Temporal Encoding

The main component of the KHP model is self-attention mechanism. Unlike RNNs, which process

data sequentially, the self-attention mechanism processes events in parallel. To account for the temporal aspect of the events, we use a temporal encoding method similar to the positional encoding in the original Transformer [4]:

$$[z(t_j)]_i = \begin{cases} \cos\left(\frac{t_j}{10000^{\frac{i-1}{M}}}\right) & \text{if } i \% 2 == 1, \\ \sin\left(\frac{t_j}{10000^{\frac{i}{M}}}\right) & \text{otherwise.} \end{cases}$$

This encoding process transforms each timestamp  $t_j$  into a fixed-dimensional vector  $z(t_j) \in \mathbb{R}^M$ . Additionally, an embedding matrix  $U \in \mathbb{R}^{M \times K}$  is trained for the event types, where each column represents a distinct event type embedding. Given an event sequence  $\mathcal{S} = \{(t_j, k_j)\}_{j=1}^L$ , the embedding is computed as:

$$X = (UY + Z)^T,$$

where  $Y = [k_1, k_2, \dots, k_L] \in \mathbb{R}^{K \times L}$  is the matrix of one-hot encoded event types, and  $Z = [z(t_1), z(t_2), \dots, z(t_L)] \in \mathbb{R}^{M \times L}$  contains the temporal encodings. Consequently, the resulting matrix  $X \in \mathbb{R}^{L \times M}$  holds the embedded representations of the events in the sequence.

After the encoding and embedding layers, we pass  $X$  through the self-attention module. The attention output  $S$  is computed as follows:

$$S = \text{Softmax}\left(\frac{QK^T}{\sqrt{M_K}}\right)V,$$

The matrices  $Q = XW_Q$ ,  $K = XW_K$ , and  $V = XW_V$  are derived from different transformations of  $X$ , where  $Q$ ,  $K$ , and  $V$  represent the query, key,

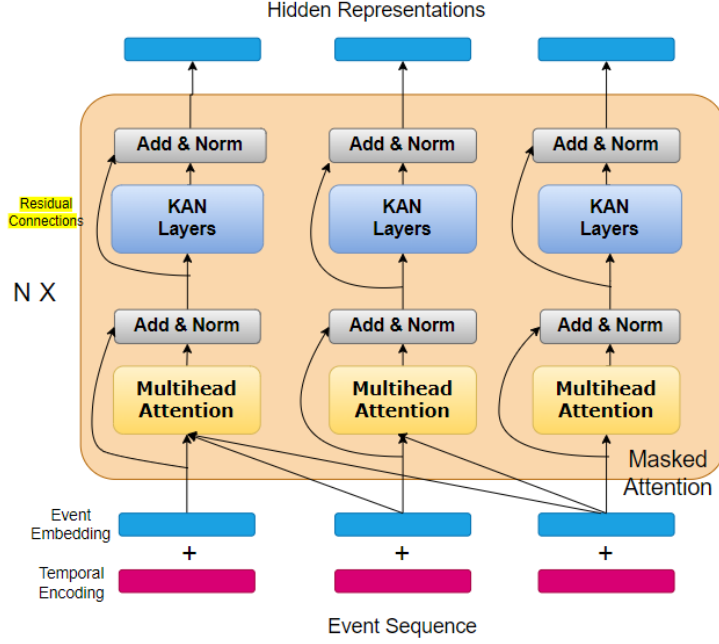


Figure 2: Encoder Block of KANformer Hawkes Process (KHP) Model

and value matrices, respectively. The weight matrices  $W_Q, W_K \in \mathbb{R}^{M \times M_K}$  and  $W_V \in \mathbb{R}^{M \times M_V}$  are utilized for these linear transformations.

To enhance the model’s flexibility, multi-head self-attention is employed. This involves computing different attention outputs  $S_1, S_2, \dots, S_H$  using distinct sets of weights  $\{W_Q^h, W_K^h, W_V^h\}_{h=1}^H$ . The final attention output for the event sequence is obtained by concatenating these outputs and applying an aggregation matrix  $W_O$  of size  $\mathbb{R}^{H M_V \times M}$ .

The self-attention mechanism enables direct selection of events irrespective of their temporal distance from the current time. Each column of the attention weights matrix  $\text{Softmax}\left(\frac{QK^T}{\sqrt{M_K}}\right)$  indicates the dependency of event  $t_j$  on its past events. In contrast, RNN-based models encode historical information sequentially through the hidden states of events, meaning the state of  $t_j$  depends on  $t_{j-1}$ ,  $t_{j-1}$  depends on  $t_{j-2}$ , and so on. If any intermediate encoding is weak, the hidden representations of subsequent events will be compromised.

To prevent the model from using future information during training, we apply masks in the attention mechanism. While computing attention output  $S_{j,:}$  for the  $j$ -th event, we mask upcoming all future positions, ensuring that  $Q_{j,j+1}, Q_{j,j+2}, \dots, Q_{j,L}$  are set to  $-\infty$ . This ensures the softmax function does not

assign attention to future events.

### 3.2 KAN Layer

In the KHP, we replace the traditional feedforward layers in the Transformer with KAN layers. These layers utilize learnable activation functions on edges, modeled as splines, enhancing the model’s capacity to capture complex patterns in the data. Each KAN layer consists of three main components: a layer normalization step, a Radial Basis Function (RBF) transformation, and a Spline Linear transformation.

**Layer Normalization** Layer normalization stabilizes and accelerates the training process. For an input  $x$ , the process is defined as:

$$\text{LN}(x) = \left( \frac{x - \mu}{\sigma} \right),$$

Here  $\mu$  and  $\sigma$  stands for mean and standard deviation of  $x$ .

**Radial Basis Function Transformation** The RBF transformation maps the normalized input into a higher-dimensional space using Gaussian kernels centered at fixed grid points:

$$\text{RBF}(x) = \exp\left(-\left(\frac{x - \mu_1}{\sigma_1}\right)^2\right),$$

where  $\mu_1$  represents the grid points and  $\sigma_1$  is a scaling parameter.

**Spline Linear Transformation** The spline linear transformation applies a linear transformation to the output of the RBF:

$$\text{SplineLinear}(x) = Wx,$$

where  $W$  is the weight matrix.

**Base Update** Optionally, a base update can be applied using a standard activation function such as SiLU and a linear transformation:

$$\text{SiLU}(x) = x \cdot \left(\frac{1}{1 + e^{-x}}\right)$$

$$\text{BaseUpdate}(x) = \text{Linear}(\text{SiLU}(x)).$$

Combining these components, hidden representations as the output of a KAN layer for an input  $S$  is:

$$\mathcal{H} = \text{SplineLinear}(\text{RBF}(\text{LN}(S))) + \text{BaseUpdate}(S).$$

### 3.3 Conditional Intensity Function

Temporal point processes are characterized by a continuous conditional intensity function [15]. In our model, the intensity function  $\lambda(t | \mathcal{H}_t)$ , where  $\mathcal{H}_t$  is the history up to time  $t$ , is given by:

$$\lambda(t | \mathcal{H}_t) = \sum_{k=1}^K \lambda_k(t | \mathcal{H}_t),$$

where each type-specific intensity is defined as:

$$\lambda_k(t | \mathcal{H}_t) = f_k\left(\alpha_k \frac{t - t_j}{t_j} + w_k^T h(t_j) + b_k\right).$$

The function  $f_k(x) = \beta_k \log(1 + \exp(x/\beta_k))$  is the softplus function, ensuring positivity and stability of the intensity values.

### 3.4 Training and Likelihood Computation

For an event sequence  $\mathcal{S}$  over the interval  $[t_1, t_L]$ , the log-likelihood is expressed as:

$$\ell(\mathcal{S}) = \sum_{j=1}^L \log \lambda(t_j | \mathcal{H}_j) - \int_{t_1}^{t_L} \lambda(t | \mathcal{H}_t) dt.$$

To optimize the model parameters, we maximize the log-likelihood across all sequences using stochastic gradient descent methods such as ADAM [7]. Techniques like layer normalization [8] and residual connections [5] are used to ensure stable training.

The integral in the non-event log-likelihood term can be approximated using Monte Carlo integration [6]:

$$\hat{\Lambda}_{MC} = \sum_{j=2}^L (t_j - t_{j-1}) \left( \frac{1}{N} \sum_{i=1}^N \lambda(u_i) \right),$$

where  $u_i \sim \text{Unif}(t_{j-1}, t_j)$ .

Alternatively, numerical integration methods such as the trapezoidal rule can be used for faster computations. The trapezoidal rule approximation for the integral is:

$$\hat{\Lambda}_{NU} = \sum_{j=2}^L \left( \frac{t_j - t_{j-1}}{2} \right) (\lambda(t_{j-1} | \mathcal{H}_{j-1}) + \lambda(t_j | \mathcal{H}_j)).$$

## 4 Experiments

We evaluate the KANformer Hawkes Process (KHP) against several baseline models: Recurrent Marked Temporal Point Process (RMTTP) [9], Neural Hawkes Process (NHP) [2], and Transformer Hawkes Process (THP) [3]. The evaluation metrics are event prediction accuracy and per-event log-likelihood (in nats) on test datasets. Training details are provided in Section 4.2.

### 4.1 Datasets

We use several datasets to evaluate the models, as summarized in Table 1. The columns from left to right represent the dataset name, number of event types, number of events with train, dev, and test splits, and minimum, average, and maximum sequence lengths.

The first four datasets (Retweets [17], MIMIC-II, StackOverflow [19], and Financial [18]) are real-world

datasets. The remaining two datasets (Conttime and Hawkes) are synthetic. Each dataset is split into train, development, and test parts in an 80-10-10 ratio. During pre-processing, sequences are padded with zeros to match the maximum sequence length within each batch.

## 4.2 Training Details

For consistency with previous studies, we use the datasets from [9] and [2]. Detailed data pre-processing and data splits in train, dev & test are described in

these papers. Training hyper-parameters are summarized in Table 2. These experiments are conducted on a NVIDIA TITAN RTX GPU with CUDA version 12.5.

To solve the integral in the log-likelihood, we use either Monte Carlo (MC) or Numerical Integration (NU) methods. For the KAN model, the hyper-parameters include model dimension ( $M$ ), key dimension ( $M_K$ ), value dimension ( $M_V$ ), hidden dimension ( $M_H$ ), learning rate (LR), batch size (BS), and the number of grids for grid extension in KAN.

Dataset	K	# Event Tokens	Train	Dev	Test	Min SL	Mean SL	Max SL
Retweets	3	2610102	2176116	215521	218465	50	$\approx 109$	264
MIMIC-II	75	2419	1952	224	243	2	$\approx 4$	33
StackOverflow	22	480414	345116	38065	97233	41	$\approx 72$	736
Financial	2	414800	298710	33190	82900	829	2074	3319
Conttime	5	602984	479467	61736	61781	20	$\approx 60$	100
Hawkes	5	602697	482947	59683	60067	20	$\approx 60$	100

Table 1: Datasets statistics.

Dataset	#Heads	#Layers	M	$M_K, M_V$	$M_H$	Dropout	#grids	BS	LR	Solver
Retweets	3	3	64	16	256	0.1	5	16	0.01	MC
MIMIC-II	3	3	64	16	256	0.1	5	1	0.0001	NU
StackOverflow	4	4	512	512	1024	0.1	5	4	0.0001	NU
Financial	6	6	128	64	2048	0.1	5	1	0.0001	NU
Conttime	3	3	64	16	256	0.1	5	1	0.01	MC
Hawkes	3	3	64	16	256	0.1	5	1	0.01	MC

Table 2: Training & testing hyper-parameters.

## 4.3 Log-Likelihood Comparison

We fit KANformer Hawkes Process (KHP) and Transformer Hawkes Process (THP) on real-world datasets MIMIC-II, Retweets, Financial and StackOverflow. We also fit our model on synthetic datasets Conttime

and Hawkes. Figure 3, 4 and 5 shows that KHP outperforms THP during the entire training process on almost every datasets. This demonstrates the ability of KHP to model the complex dynamics of social media data better than THP.

Table 3 summarizes the log-likelihood results on all datasets. Our KHP model demonstrates a strong fit to the data and surpasses the majority of baseline models.

Model	MIMIC-II	StackOverflow	Financial	Retweets	Conttime	Hawkes
RMTTPP	-1.35	-2.60	-3.89	-5.99	-	-
NHP	-1.38	-2.55	-5.60	-3.60	-	-
THP	0.40	-0.72	-0.90	-3.22	1.04	0.89
KHP (our)	<b>0.72</b>	<b>-0.70</b>	<b>-0.56</b>	<b>-2.67</b>	<b>1.09</b>	<b>0.97</b>

Table 3: Comparison of Log-likelihood.

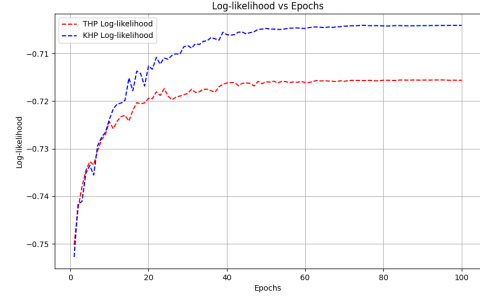
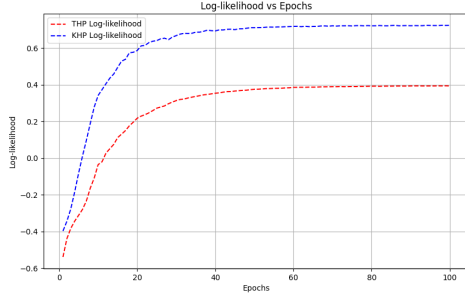


Figure 3: Comparison of testing curves between THP and KHP on MIMIC-II (left) and StackOverflow (right), focusing on log-likelihood results.

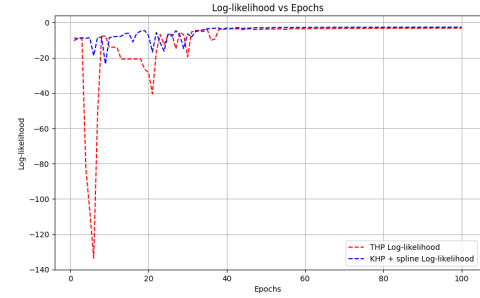
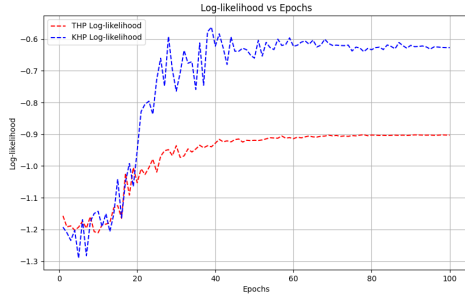


Figure 4: Comparison of testing curves between THP and KHP on Financial (left) and Retweets (right), focusing on log-likelihood results.

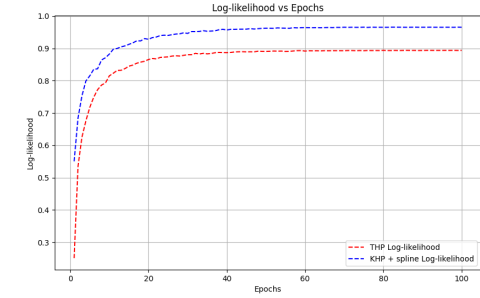
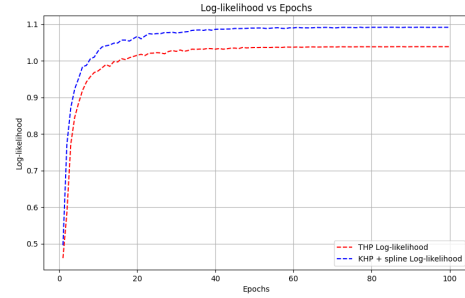


Figure 5: Comparison of testing curves between THP and KHP on Conttime (left) and Hawkes (right), focusing on log-likelihood results.

#### 4.4 Event Prediction Comparison

Accurately predicting future events and their timings is essential for point process models. By adding prediction layers to the KHP model, we can improve its performance. The following equations detail the methods for predicting the next event type and time:

- Next event type prediction given as:

$$\hat{k}_{j+1} = \arg \max_k \hat{p}_{j+1}(k),$$

$$\hat{p}_{j+1} = \text{Softmax}(\mathbf{W}^{\text{type}} h(t_j)),$$

where  $\mathbf{W}^{\text{type}} \in \mathbb{R}^{K \times M}$  is the weight matrix for the type predictor.



- Next event time prediction given as:

$$\hat{t}_{j+1} = \mathbf{W}^{\text{time}} h(t_j),$$

where  $\mathbf{W}^{\text{time}} \in \mathbb{R}^{1 \times M}$  is the weight matrix for the time predictor.

The loss function comprises a mean squared error term for event time prediction and a cross-entropy term for event type prediction. For the event sequence  $\mathcal{S} = \{(t_j, k_j)\}_{j=1}^L$ , the loss terms are defined as:

$$\mathcal{L}_{\text{type}}(\mathcal{S}) = \sum_{j=2}^L -k_j^\top \log(\hat{p}_j),$$

$$\mathcal{L}_{\text{time}}(\mathcal{S}) = \sum_{j=2}^L (t_j - \hat{t}_j)^2.$$

The total loss function is:

$$\mathcal{L} = \sum_{i=1}^N -\ell(\mathcal{S}_i) + \mathcal{L}_{\text{type}}(\mathcal{S}_i) + \mathcal{L}_{\text{time}}(\mathcal{S}_i),$$

Below are the curves of event type prediction accuracy and event time prediction rmse on test datasets.

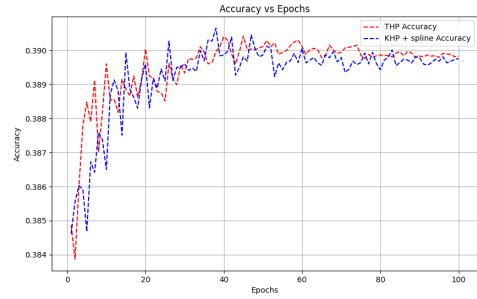
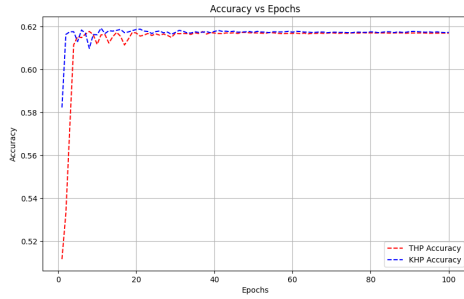


Figure 6: Comparison of testing curves between THP and KHP on Financial (left) and Conttime (right), focusing on event type prediction accuracy results.

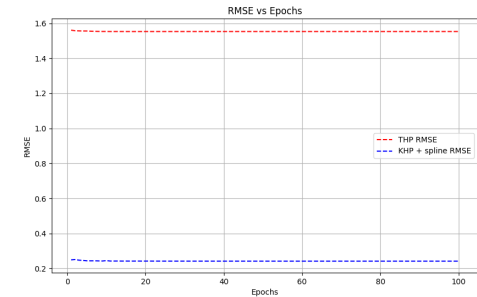
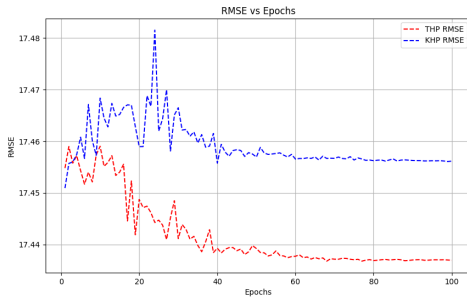


Figure 7: Comparison of testing curves between THP and KHP on Financial (left) and Conttime (right), focusing on event time prediction rmse results.

where  $\ell(\mathcal{S}_i)$  represents log-likelihood of sequence  $\mathcal{S}_i$ .

For event time prediction, the RMSE is calculated as:

$$\text{RMSE} = \sqrt{\frac{1}{L} \sum_{j=1}^L (t_j - \hat{t}_j)^2},$$

where  $t_j$  is the actual event time and  $\hat{t}_j$  is the forecasted event time.

For event type prediction, the accuracy is defined as:

$$\text{Accuracy} = \frac{1}{L} \sum_{j=1}^L \mathbb{I}(k_j = \hat{k}_j),$$

where  $k_j$  represents the actual event type,  $\hat{k}_j$  is the forecasted event type, and  $\mathbb{I}$  is an indicator function that returns 1 if the specified condition is met and 0 otherwise.

Tables 4 and 5 present the event prediction results, showing that KHP generally outperforms the baseline models in most tasks.

Model	Financial	Retweets	Conttime	Hawkes
RMTTPP	61.95	51.97	-	-
NHP	<b>62.20</b>	53.67	-	-
THP	61.77	<b>54.05</b>	39.04	38.13
KHP	61.93	53.91	<b>39.06</b>	<b>38.14</b>

Table 4: Comparison of Event type prediction Accuracy (%).

Model	Financial	Retweets	Conttime	Hawkes
RMTTPP	29.27	36613.23	-	-
NHP	29.27	36013.65	-	-
THP	17.45	<b>35126.29</b>	1.55	2.29
KHP	<b>17.43</b>	35149.99	<b>0.24</b>	<b>0.43</b>

Table 5: Comparison of Event time prediction RMSE.

## 5 Conclusion

The KANformer Hawkes Process (KHP) introduces an innovative approach to analyzing event sequences by combining the Transformer architecture with Kolmogorov-Arnold Networks (KANs). By integrating self-attention mechanisms with learnable activation functions, KHP efficiently captures both short-term dependencies and long-term dependencies in event data while maintaining computational efficiency. Empirical evaluations on various real-world datasets reveal KHP’s superior performance in terms of event prediction accuracy and log-likelihood. The model’s flexibility and its capability to incorporate structural knowledge position it as a valuable tool for analyzing complex event streams across diverse domains. Future research directions include extending the KAN framework to other types of neural network architectures, exploring its application in multi-modal data analysis, and enhancing the model’s interpretability and robustness to handle missing or noisy data, further expanding its utility in real-world scenarios.

## References

- [1] Liu, Ziming, et al. "Kan: Kolmogorov-arnold networks." arXiv preprint arXiv:2404.19756 (2024).
- [2] Mei, H., & Eisner, J. (2017). The Neural Hawkes Process: A Neurally Self-Modulating Multivariate Point Process. *Advances in Neural Information Processing Systems*, 30.
- [3] Zuo, Simiao, et al. "Transformer hawkes process." International conference on machine learning. PMLR, 2020.
- [4] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention Is All You Need. *Advances in Neural Information Processing Systems*, 30.
- [5] He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.
- [6] Robert, Christian P., George Casella, and George Casella. Monte Carlo statistical methods. Vol. 2. New York: Springer, 1999.
- [7] Kingma, Diederik P., and Jimmy Ba. "Adam: A method for stochastic optimization." arXiv preprint arXiv:1412.6980 (2014).
- [8] Ba, Jimmy Lei, Jamie Ryan Kiros, and Geoffrey E. Hinton. "Layer normalization." arXiv preprint arXiv:1607.06450 (2016).
- [9] Du, Nan, et al. "Recurrent marked temporal point processes: Embedding event history to vector." Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining. 2016.
- [10] Cho, Kyunghyun, et al. "Learning phrase representations using RNN encoder-decoder for

- statistical machine translation." arXiv preprint arXiv:1406.1078 (2014).
- [11] Dey, Rahul, and Fathi M. Salem. "Gate-variants of gated recurrent unit (GRU) neural networks." 2017 IEEE 60th international midwest symposium on circuits and systems (MWSCAS). IEEE, 2017.
  - [12] Greff, Klaus, et al. "LSTM: A search space odyssey." IEEE transactions on neural networks and learning systems 28.10 (2016): 2222-2232.
  - [13] Hawkes, Alan G. "Hawkes processes and their applications to finance: a review." Quantitative Finance 18.2 (2018): 193-198.
  - [14] Kingman, John Frank Charles. Poisson processes. Vol. 3. Clarendon Press, 1992.
  - [15] Bradley, Brendon A. "A ground motion selection algorithm based on the generalized conditional intensity measure approach." Soil Dynamics and Earthquake Engineering 40 (2012): 48-61.
  - [16] SS, Sidharth. "Chebyshev Polynomial-Based Kolmogorov-Arnold Networks: An Efficient Architecture for Nonlinear Function Approximation." arXiv preprint arXiv:2405.07200 (2024).
  - [17] Zhang, Zizhu, et al. "A study on the retweeting behaviour of marketing microblogs with high retweets in Sina Weibo." 2015 Third International Conference on Advanced Cloud and Big Data. IEEE, 2015.
  - [18] Hemmelgarn, Thomas, et al. "Financial transaction taxes in the European Union." National Tax Journal 69.1 (2016): 217-240.
  - [19] Kou, Bonan, et al. "Sosum: A dataset of stack overflow post summaries." Proceedings of the 19th International Conference on Mining Software Repositories. 2022.
  - [20] Du, Nan, et al. "Recurrent marked temporal point processes: Embedding event history to vector." Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining. 2016.
  - [21] Oreshkin, Boris N., et al. "N-BEATS: Neural basis expansion analysis for interpretable time series forecasting." arXiv preprint arXiv:1905.10437 (2019).
  - [22] Hawkes, Alan G. "Hawkes processes and their applications to finance: a review." Quantitative Finance 18.2 (2018): 193-198.
  - [23] González, Jonatan A., et al. "Spatio-temporal point process statistics: a review." Spatial Statistics 18 (2016): 505-544.
  - [24] Omi, Takahiro, and Kazuyuki Aihara. "Fully neural network based model for general temporal point processes." Advances in neural information processing systems 32 (2019).
  - [25] Challu, Cristian, et al. "Nhits: Neural hierarchical interpolation for time series forecasting." Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 37. No. 6. 2023.
  - [26] Lüdke, David, et al. "Add and thin: Diffusion for temporal point processes." Advances in Neural Information Processing Systems 36 (2024).
  - [27] Beal, George M., and Joe M. Bohlen. "The diffusion process." (1956): 111-121.

## Acknowledgement

I express my deepest gratitude to my project advisor, **Dr. Ambedkar Dukkipati**, for his unwavering support and invaluable guidance throughout this project. Dr. Dukkipati's expertise in machine learning and event stream modeling has been instrumental in navigating the complex challenges and refining the methodologies employed in this research. His meticulous feedback, constructive criticism, and willingness to engage in thoughtful discussions have significantly enhanced my understanding and improved the quality of this project. Dr. Dukkipati's commitment to academic excellence and personal encouragement have been immensely motivating, giving me the confidence to pursue ambitious goals. I am profoundly grateful for the opportunity to work under his supervision and for the significant impact he has had on my academic and professional development. Thank you, Dr. Dukkipati, for your exceptional mentorship.