1 Abstract

Many real-world sequential datasets are irregularly sampled and characterized by a mixture of numerical and categorical features. Such event sequences are critical in domains like healthcare, physics, social media, and finance, where observations arrive continuously over time. Traditional approaches—most notably recurrent neural networks (RNNs)—often discretize time by binning or imputing timestamps, thereby neglecting the underlying continuous dynamics or discarding valuable temporal information. Neural Ordinary Differential Equations (Neural ODEs) offer a principled framework for modeling data in continuous time.

In this thesis, we focus on the unsupervised task of learning embeddings for a marked point process, where each event is assigned a discrete label. We extend the methodology first proposed by [8] in two key ways. First, we replace simple aggregation schemes with a Transformer-based module to capture long-range dependencies more effectively. Second, we introduce a decoupled architecture in which events of each type are processed by separate Neural Controlled Differential Equation (Neural CDE) systems. This design allows the model to accumulate influence at the level of event types—rather than treating every occurrence in isolation—thereby enhancing both representational capacity and interpretability.

2 Introduction

The growth of data in various domains, time series included, raises important questions about how to analyze and predict these observations.

Traditionally, time series have been assumed to be regularly sampled or have been forced into a discrete framework by binning or timestamp imputation. Models such as RNNs, LSTMs, GRUs, and Transformers rely on this discretization to process sequential data [6].

Neural Differential Equations offer an alternative by modeling the sequence as a continuous-in-time process via a learned dynamics function [1]. Their one drawback is that, being formulated as an initial-value problem, they cannot directly account for new events that arrive after the initial state has been set.

An extension of Neural ODEs was proposed in Neural CDEs [3], where the authors address this limitation by treating the observed event stream as a control signal that continuously perturbs the hidden trajectory, thereby allowing the model to incorporate each new event as it occurs.

Temporal Point Processes (TPPs) [7] differ from standard time series because they are

stochastic processes whose realizations consist of a sequence of discrete, time-ordered events. Predicting a TPP means estimating the time of the next event given the history of past event times and any associated information. A classical model for this task is the Hawkes process [4], which represents the stochastic behavior through a *conditional intensity function* defined as the sum of influence kernels from each previous event.

Traditional TPP models often overlook additional data that may accompany each event, such as its type, location, or other features. To capture this extra information, one considers Marked Temporal Point Processes, in which each event carries a discrete *mark* indicating its category or attributes. Recent neural approaches extend the Hawkes framework by parameterizing the intensity or update rules with neural networks [8]. Examples include RNN-based marked TPPs [2], the Neural Hawkes Process [5], and the Transformer Hawkes Process [9]. While these methods can flexibly model complex dependencies, they share two main drawbacks: (1) limited interpretability, since it is unclear how individual past events contribute to future predictions, and (2) computational difficulty in evaluating the probability density and intensity functions, often requiring additional numerical integration steps.

A recent work, Neural ODE with Decoupled Marked Temporal Point Processes [8], addresses the limitations of earlier models. In their approach, each event's influence on future dynamics is handled separately: the event's feature vector is first propagated through a Neural ODE to produce an event-specific hidden state, and these states are then linearly aggregated into a single embedding. This embedding is used to predict the event likelihood (pdf), the conditional intensity function, and to support downstream tasks.

In this thesis, we extend the [8] framework in two key ways. First, we replace the simple linear aggregation with a Transformer- and RNN- based module, which can capture more complex interactions and long-range dependencies among event embeddings. Second, we propose that events of the same type should be modeled as a collective control signal rather than as independent occurrences. To this end, we adopt Neural Controlled Differential Equations (Neural CDEs) in place of Neural ODEs, enabling each event type to drive its own latent trajectory and thus accumulate type-specific influences over time.

3 Theoretical Background

3.1 Neural ODE

In the original Neural ODE framework [1], the continuous evolution of a hidden state $\mathbf{h}(t)$ is parameterized by an ordinary differential equation, approximated by a neural network:

$$\frac{d\mathbf{h}(t)}{dt} = f(\mathbf{h}(t), t, \theta)$$

Given an initial state $\mathbf{h}(t_0)$, the final state at time T is given by:

$$\mathbf{h}(T) = \mathbf{h}(t_0) + \int_{t_0}^{T} f(\mathbf{h}(t), t, \theta)dt$$

One can think of the NeuralODE models as a continuous-depth analogue of residual neural networks: depth in residual has its continious analogy: time in ODE. The comparison is illustrated in the figure 1.

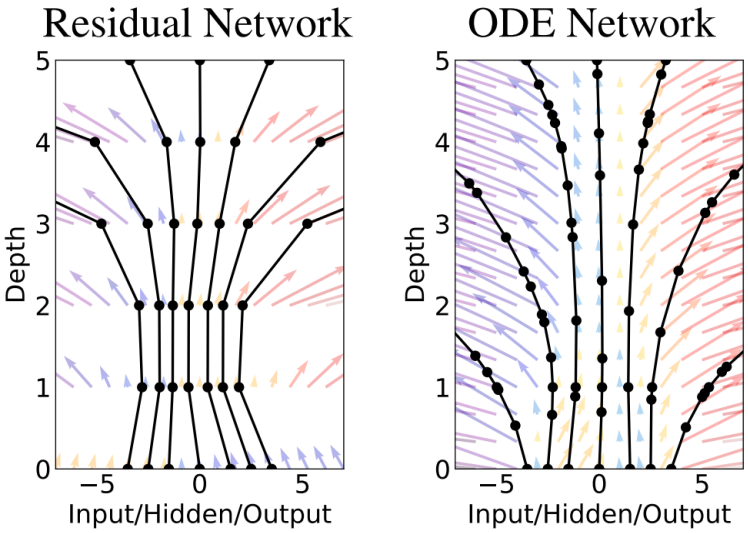

Figure 1: Residual (left) vs ODE (right) neural network [1]

In Neural ODE setting we assume that we know the $\mathbf{h}(t_0)$ and need to evaluate the output of the Neural ODE, i.e. the $\mathbf{h}(T)$.

In the same article authors suggest to use NeuralODE as a generative time-series model for irregularly-sampled data, what is closer to our task. In this setting each time series represented by its latent trajectory: $\mathbf{z_t}$. So, having an initial observation $\mathbf{z}_{t_0}$ and a set of times, at which we need to evaluate our time series $t_1, \ldots, t_N$ we can define generative model [1]:

$$\begin{cases} \mathbf{z}_{t_0} \sim p(\mathbf{z}_{t_0}) \\ \mathbf{z}_{t_i} = \text{ODESolve}(\mathbf{z}_{t_0}, f_\theta, t_i) \\ \mathbf{x}_{t_i} \sim p(\mathbf{x}|\mathbf{z}_{t_i}, \phi) \end{cases}$$

Where ODESolve is any ODE solver, $\mathbf{x}_{t_i}$ — value, corresponding to the $\mathbf{z}_{t_i}$, f_θ — time-invariant hidden dynamics model: $f_\theta(\mathbf{z}(t)) = \frac{\partial z(t)}{\partial t}$, which is a neural net with parameter θ, ϕ — parameter of decoder. So, this generative model allows us to uniquely (as f is time-invariant) reconstruct and extend any trajectory backward or forward in time. The training process illustrated in the figure 2.

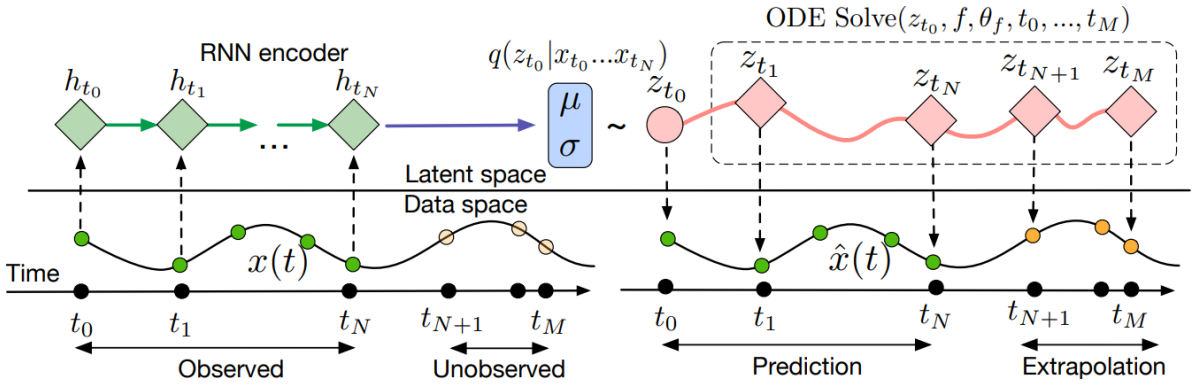

Figure 2: Neural ODE training for time series

As one can see, authors use RNN as an encoder of the whole sequence, which outputs the parameters of the distribution $q(\mathbf{z_{t_0}}|\mathbf{x}_{t_0}, \ldots, \mathbf{x}_{t_N})$, after which the prediction and extrapolation is done using an ODE Solver.

3.2 Neural CDE

One of the main drawbacks of the Neural ODE approach of prediction of time-series data is that solution to ODE is determined by its initial condition and can not be adjusted, based on the future observations. Authors of the paper Neural Controlled ODE for Irregular Time Series [3] suggest their method of *controlled differential equations* or NeuralCDE — a continious analogue of RNN. It naturally employs the arriving data.

Let's denote arriving data as $\mathbf{x} = ((t_0, x_0), (t_1, x_1), \ldots, (t_n, x_n))$, where t_i — is a time of the observation $x_i \in \mathbb{R}^v$, $t_1 < \cdots < t_n$. Then authors denote $X : [t_0, t_n] \to \mathbb{R}^{v+1}$ to be the *natural cubic spline*, such that $X_i = (x_i, t_i)$. In other words, the $\mathbf{x}$ is a discretization of the process X. Let $f_\theta : \mathbb{R}^w \to \mathbb{R}^{w \times (v+1)}$ — be a neural network (analogue vector field in ODE),

where w is a hidden state size. Let also be $l_\theta : \mathbb{R}^{v+1} \to \mathbb{R}^w$ — encoder for the observations $\mathbf{x}_i$. Then authors [3] define the Neural CDE model as the solutions of CDE:

$$\begin{cases} h_t = h_{t_0} + \int_{t_0}^{t} f_\theta(h_t) dX_t, \quad \text{for} t \in (t_0, t_n] \\ l_\theta(x_0, t_0) = h_{t_0} \end{cases}$$

As we can see, this is a natural extension of the ordinary ODE model (we just integrate along some path X). And if this path is identical $i : \mathbb{R} \to \mathbb{R}$ then we get Neural ODE.

Due to using the cubic spline, the evaluation of Neural CDE is pretty simple:

$$h_t = h_{t_0} + \int_{t_0}^{t} f_\theta(h_t) dX_t = h_{t_0} + \int_{t_0}^{t} f_\theta(h_t) \frac{dX}{dt}(t) dt$$

And can be solved with any ODE solver.

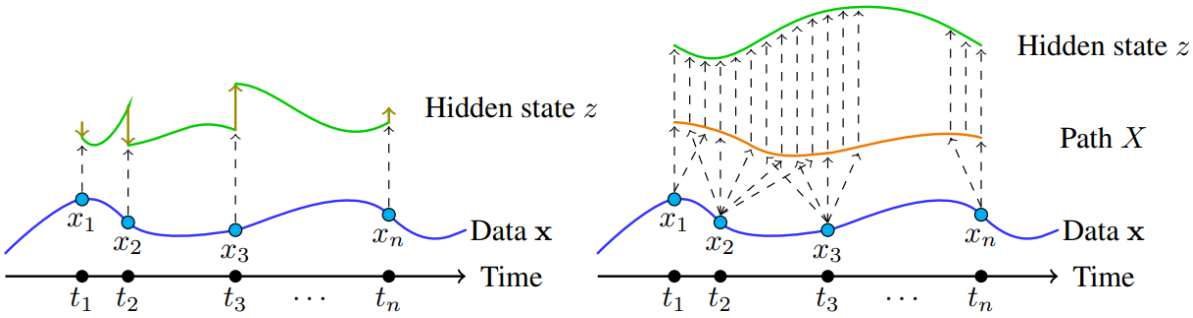

Figure 3: Neural CDE approach (right) compared to previous approaches (left) when feeding with incoming data [3]

As we can see in the figure 3, the data path X allows us to integrate the smooth change of the observed data throughout time (right), unlike other approaches (left), which modify the hidden state discretely in time only at observed timestamps.

3.3 DMTPP-ODE

Now let's look at event-time data. In this setting we have a sequence of events, happening in time. Each event is characterised by its *time* and *type*. And we need to predict, of evaluate the probability, of the next event time and its type.

For example, based on the history of purchases, one need to predict the time and type of the next purchase. In other words, we need to predict the likelihood of the future event e at specific time t, i.e. pdf $f(e, t)$ [8].

Historically, the first approach to predict this events called **Temporal Point Process (TPP)**. It is a stochastic process, whise realization is a set of ordered times $[t_0, \dots, t_N]$.

Let us denote pdf of the next event, given a set of previous events $\mathcal{H}_{t_i} = [t_0, \dots, t_i]$ as $f(t|\mathcal{H}_{t_i}) = f^*(t)$, its cdf as $F^*(t) = F(t|\mathcal{H}_{t_i})$ and a survival function $S^*(t) = S(t|\mathcal{H}_{t_i}) = 1 - F(t|\mathcal{H}_{t_i})$. As f^* is constained as a probability measure function, it is hard to parametrize. Instead, the authors suggest to use the *intensity function* $\lambda^*(t) = \frac{f^*(t)}{S^*(t)}$ with no constraints except the non-negativeness [7]. Its meaning is defined as follows:

$$\lambda^*(t)dt = p(t_i \in [t, t + dt)|\mathcal{H}_t)$$

The approach, which was suggested in DMTPP-ODE paper [8] for prediction of future events and their classes has its foundations on the decoupling of the effects of past events. So, each event has a separate influence on the pdf function of a future event — a technique, which has a connection with Hawkes Processes [4], where the intensity function is aggregates influence from past events by the following way:

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

Where $\mu(t)$ is some decreasing function, that models the effect, which decaying with time.

The detailed explanation of the DP TPP is presented on the figure 4.

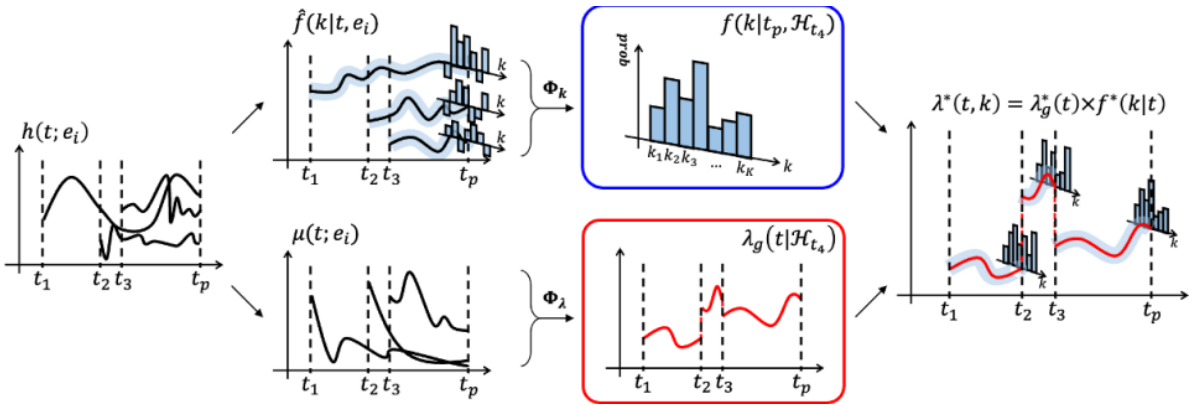

Figure 4: DM TPP scheme [8]

Each event type $k \in [1, \dots, K]$ has a separate trainable embedding $W_e(k)$. For each event $e_i = (t_i, k_i)$ its embedding at its starting point $h(t_i; e_i) = W_e(k_i)$ is propagated with hidden dynamics γ_θ (parametrized by neural network).

$$dh(t; e_i) = \gamma_\theta(h(t; e_i), t, k_i)dt$$

And then influence the final *ground intensity function* separately:

$$\lambda_g(t|\mathcal{H}_{t_{n+1}}) = \lambda_g(t|e_0, \dots, e_n) = \mathsf{A}(g_\mu(h(t; e_n), g_\mu(h(t; e_n), \dots, g_\mu(h(t; e_n))$$

Where the function $g_\mu : \mathbb{R}^d \to \mathbb{R}$ is a neural network, modeling influence function and A is some non-negative aggregation function, for example, softplus.

They analougously influence the conditional probability for marks $f^*(k|t)$:

$$f(k|t, \mathcal{H}_{t_{n+1}}) = \mathsf{A}_k(g_f(h(t; e_1)), g_f(h(t; e_2)), \ldots, g_f(h(t; e_n)))$$

Where $g_f : \mathbb{R}^d \to \mathbb{R}$ and A_k satisfy: $\sum_{k=1}^{K} \mathsf{A}_k(\cdot) = 1$, and can be a softmax function.

The DM-TPP is trained by optimizing log-likelihood. Since each event has a separate influence, the likelihood is defined as follows:

$$L = \left[\prod_{i=1}^{N} \lambda_g^*(t_i)\right] \cdot \left[\prod_{i=1}^{N} f^*(k_i|t_i)\right] \cdot exp(-\int_0^{t_N} \lambda_g^*(u)du)$$

And the corresponding log-llikelihood is:

$$\ln(L) = \sum_{i=1}^{N} \ln(\lambda_g^*(t_i)) - \int_0^{t_N} \lambda_g^*(u)du + \sum_{i=1}^{N} \ln(f^*(k_i|t_i))$$

3.4 DMTPP-ODE with Transfomer

Here we will use the transfomer encoder architecture to combine embeddings from different events, instead of simple linear aggregation:

$$\lambda_g(t|\mathcal{H}_{t_{n+1}}) = \lambda_g(t|e_0, \ldots, e_n) = g_\mu(\text{Transformer}(h(t; e_n), h(t; e_n), \ldots, h(t; e_n)))$$

$$f(k|t, \mathcal{H}_{t_{n+1}}) = g_f(\text{Transformer}(h(t; e_1), h(t; e_2), \ldots, h(t; e_n)))_k$$

3.5 DMTPP-CDE

4 Approach

5 Problem statement

We consider a problem of supervised TPP classification task.

Given a MTPP (Marked Time Point Process) $\mathcal{S} \in \mathbb{S}$, $\mathcal{S} = \{e_i\}_{i=0}^N = \{(t_i, k_i)\}_{i=0}^N$; $0 \leq t_i < t_{i+1} \leq 1$; $k_i \in [1, \ldots, K]$, and its label $y \in \{0, 1\}$. Where K is the total number of event types, t_i is a time of event. $\mathbb{S}$ is a set of all MTPPs.

We choose a classification model, parametrized by a neural network, $h_\theta : \mathbb{S} \to [0, 1]$ to minimize cross-entropy loss L:

$$\theta^* = \arg\min_\theta (\sum L(h_\theta(\mathcal{S}_i), y_i))$$

6 Experiments

We used Churn dataset. Each item is a time series of the mean length of 100. Each evens in characterised by one of the 10 types. We tried to predict the probability of churn of the client based on his actions.

We tried to use three aggregation techniques: *mean*, *LSTM* and *transformer* aggregation. We measured the accuracy and ROC-AUC and measures results on validation dataset.

The results are presented in Table 1 (they were averaged over 5 runs).

Table 1: Experiment results. Mean and 3 standart deviations

Aggregation	Accuracy	ROC AUC
Mean	$69.2\% \pm 0.9\%$	$71.4\% \pm 0.8\%$
LSTM	$74.0\% \pm 2.0\%$	$77.5\% \pm 2.6\%$
Transformer	$70.3\% \pm 3.0\%$	$71.6\% \pm 5.0\%$

Conclusions

1. Default aggregation (mean) showed the worst results.
2. LSTM scored best on both Accuracy and ROC AUC.
3. Transformer model seemed to be undertrained, since it has a the biggest discrepance.

References

[1] Ricky TQ Chen et al. "Neural ordinary differential equations". In: *Advances in neural information processing systems* 31 (2018).

[2] Nan Du et al. "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*. 2016, pp. 1555–1564.

[3] Patrick Kidger et al. "Neural controlled differential equations for irregular time series". In: *Advances in neural information processing systems* 33 (2020), pp. 6696–6707.

[4] Hongyuan Mei and Jason M Eisner. "The neural hawkes process: A neurally self-modulating multivariate point process". In: *Advances in neural information processing systems* 30 (2017).

[5] Hongyuan Mei and Jason M Eisner. "The neural hawkes process: A neurally self-modulating multivariate point process". In: *Advances in neural information processing systems* 30 (2017).

[6] YongKyung Oh et al. "Comprehensive Review of Neural Differential Equations for Time Series Analysis". In: *arXiv preprint arXiv:2502.09885* (2025).

[7] Jakob Gulddahl Rasmussen. "Lecture notes: Temporal point processes and the conditional intensity function". In: *arXiv preprint arXiv:1806.00221* (2018).

[8] Yujee Song et al. "Decoupled marked temporal point process using neural ordinary differential equations". In: *arXiv preprint arXiv:2406.06149* (2024).

[9] Simiao Zuo et al. "Transformer hawkes process". In: *International conference on machine learning*. PMLR. 2020, pp. 11692–11702.