Learning Generative RNN-ODE for Collaborative Time-Series and Event Sequence Forecasting

Longyuan Li, Yunhao Zhang, Jihai Zhang, Junchi Yan, *Member, IEEE*
Yaohui Jin and Xiaokang Yang, *Fellow, IEEE*

Abstract—Time-series and event sequences are widely collected data types in many applications. Analysis and prediction of them play an important role in the decision-making process. A major limitation of previous methods is that they either focus on continuous time series or discrete events, rather than the combination of the two types of data, ignoring the correlation between them. In this paper, we consider the problem of joint modeling and forecasting of time-series and event sequence. However, the two types of data provide complementary information for the temporal dynamics, emphasizing the necessity of jointly modeling the both. We propose the RNN-ODE collaborative model for joint modeling and forecasting of heterogeneous time-series and event sequence data, which combines several. To learn complex correlations across heterogeneous sequences, we devise a tailored encoder to combine the advances in deep point processes models and variational recurrent neural networks. To predict the probability of event occurrence at arbitrary continuous-time horizon, we leverage the mathematical foundation of novel Neural Ordinary Differential Equations (NODE). It is proved on multiple simulation and real data sets that compared with existing methods, integrated modeling and prediction can effectively extract features and improve the prediction performance of time series and event sequences.

Index Terms—Probabilistic Forecasting, Event Prediction, Temporal Point Processes, Time-Series, Variational Auto-Encoder, Ordinary Differential Equations

1 Introduction

Nowadays, temporal data has been widely collected and analyzed across different areas including social networks, electronic health, economics, financial market, transportation. According to different collection methods, there are mainly two types of data: time-series and event sequence. Ideally, time-series are produced by observing variables that change continuously over time. In practice, by sampling or aggregating at fixed intervals, the time-series data are usually represented in the form of evenly-sampled discrete-time sequences, also called synchronous sequence [1]. Typical time-series data include sensor data, economic index, and traffic flow data, etc. On the other hand, event sequences are also called asynchronous sequences, which consist of two-tuples (event type and event occur time) that are irregularly dispersed in the continuous-time domain [2]. Typical event sequence data refers to electrical heal records (EHR), e-commerce transactions, social network interactions, etc. Modeling and forecasting of the two types of temporal data play an important role in the decision making process in many applications, which is also the main focus of this paper.

There have been recent works on modeling time-series [3], [4], [5], [6], [7], [8] and event sequence [9], [10], [11], [2], [12]. In fact, these methods deal with time-series and event sequences independently, ignoring the dependence between them. However, in real-world applications, it is known that time-series and event sequences are strong correlated [1]. In other words, external events affect time-series frequently, and certain patterns in time-series may also drive events. For example, in health care applications, medication intake affect blood pressure, and long-term high blood pressure may cause heart attack.

As discussed, time-series and event sequences are usually strongly correlated and provide complementary information. Therefore, collaboratively modeling the two types of temporal sequence becomes an urgent need. However, as far as we know, there is currently a lack of principled integrated methodologies for such heterogeneous sequences, which in fact pose unique challenges: the time-series are continuously observed in the discrete-time domain at fixed intervals, while the events irregularly occur in the continuous-time domain. Such misalignment cannot be directly well handled by most existing machine learning methods. There are related efforts in integrating the time-series and event sequences. One way is to extract events from time-series data according to human-defined rules [10]. However, it is somehow ad-hoc to define the event detection model or rules. Oppositely, the other way is to aggregate event counts at a fixed interval to extract aligned time-series data. However, such a coarse processing method not only causes key information loss about the actual behavior of the process but also leads to sparse time-series that are difficult to learn [13].

Seeing the above issues, in this paper, we aim to address the problem of integrated modeling and forecasting of time-series and event sequences. To the best of our knowledge,

L. Li and Y. Jin are with State Key Lab of Advanced Optical Communication System and Network, and MoE Key Lab of Artificial Intelligence, AI Institute, Shanghai Jiao Tong University, Shanghai, 200240, P.R. China.
Y. Zhang, J. Zhang, and J. Yan are with Department of Computer Science and Engineering, and MoE Key Lab of Artificial Intelligence, AI Institute, Shanghai Jiao Tong University, Shanghai, 200240, P.R. China.
{jeffli,zhangyunhao,yunfan243332345,yanjunchi, jinyh}@sjtu.edu.cn
Junchi Yan and Yaohui Jin are the corresponding authors.

this is the first work that considers this problem[1]. This is particularly challenging for the following reasons: 1) The joint timeline of time-series and event sequence is a mixture of discrete and continuous timelines, which makes it incompatible with existing time-series and event prediction models. 2) Time-series and event sequences are complex non-stationary, multi-modal random processes, making them difficult to predict, especially for long-term predictions. The model is required to describe the usefulness and limitation of the prediction results through uncertainty modeling. 3) The correlation between events and time-series is heterogeneous, i.e., there exist different types of events, multiple dimensions of time-series, and different relationships between them. 4) Within the forecast horizon, events may occur arbitrarily at any continuous-time points, instead of a fixed number of times.

To tackle the above challenges, first, we transform the forecasting problem into a conditional probability density estimation problem. Then we propose a model based on the conditional VAE framework to optimize the edge probability density function and derive a principled objective function to learn the highly nonlinear generative model. Specifically, we deal with the non-aligned timeline by carefully devising a Time-Aware Fusion RNN component as the embodiment of both the recognition model and prior model to for heterogeneous sequence features learning. Moreover, we present a novel Neural Ordinary Differential Equation (Neural ODE) based event decoder to learn a continuous function of time $f(t)$ that represents the likelihood of event occurrence at t. Such a design allows for different numbers of events that may occur in the forecasting horizon. In particular, different from previous event prediction models that could only output a single prediction (e.g. the most likely next event), we model the evolution of the event occurrence likelihood over time, which is more informative and coherent to the downstream decision-making tasks.

The contribution of the paper can be summarized as follows.

- We formally define the long-term temporal forecasting problem: joint prediction of event sequences over a fixed given time window, together with the forecasting of time series over multiple horizons. This is in contrast to most existing works either focused on time series forecasting or event prediction.
- Under the conditional VAE framework, we propose a novel conditional density estimation model to optimize the edge density and derive the objective to learn the generative temporal model. Its probabilistic nature allows for flexible uncertainty modeling over time which is of vital importance to long-term forecasting for both events and time series.
- For event prediction performance evaluation of our probabilistic model, we devise a clustering based technique to convert sampling on the intensity function of temporal point process into a deterministic output, which is still an open problem in literature.
- Experimental results including ablation studies show the effectiveness of our devised components, including the time-aware hybrid RNN encoder, neural ODE decoder, and auto-regressive time-series decoder. In particular, our method outperforms state-of-the-art peer methods in accuracy for both time-series forecasting and probabilistic event prediction tasks.

1. A preliminary part (for time series forecasting only) of this article has been published in the conference paper [14]. In this journal version, we have rewritten the paper (including the redrawn figures) and made significant technical extensions including: i) We extend and formulate the problem to a more challenging setting i.e. forecasting event sequence given a future time window, which is often the case for practical decision making. However, existing literature mostly focus on next event prediction, and fixed-window event forecasting has been rarely studied, let alone for its general case regarding with different event types; ii) To fulfill the window-based event forecasting, we propose a neural ODE based generative model for event sequence forecasting which can be jointly trained with the other parts of the whole model in an end-to-end differentiable fashion. Moreover, we devise a clustering based technique to convert probabilistic sampling on the intensity function into a deterministic estimation to enable quantitative evaluation of event sequence forecasting, which is still an open problem in literature; iii) We conduct comprehensive experiments on multiple datasets to show the promising results of our forecasting model, which further outperforms the conference version [14] for time series forecasting. Moreover, detailed case studies are given to illustrate how the period-based event sequence forecasting can be fulfilled and evaluated.

The rest of the paper is organized as follows. Section 2 reviews event prediction and time-series forecasting methods which are found relatively separated into two independent lines of research. In Section 3, we introduce the preliminaries of the proposed model. In Section 4, we present our integrated model of heterogeneous sequences. In Section 5, we show results on multiple challenging datasets. Finally, concluding remarks are given in Section 6.

2 Related Works

Depending on the way the data is generated, there are two main types of temporal data: time-series and event sequence, both call for effective forecasting over time. For the time-series data, the observations are sampled from continuously changed variables, where the sampling interval is fixed. For event sequence, observation is only recorded when an event occurs. Due to the correlation between events, the occurrence of an event may stimulate or suppress the probability of the next event. For example, in social networks, the interaction between two users may inspire more interactions. In this case, the timing and interval of events carry rich information about the dynamics. Because of the different characteristics of temporal data, existing literature works attempt to solve the prediction problem in two orthogonal categorical of methods.

2.1 Event Prediction

Temporal Point Process (TPP) is a sophisticated framework for modeling a sequence of events in continuous time domain [15], [16]. It directly estimate the rate of event occurrence, by using a random process whose realization involves a list of discrete events localized in time. The dynamics of point process can be represented by its conditional intensity function. Traditional TPP methods such as self-exciting processes are widely used for modeling the dynamics of earthquake aftershocks, news-feed streams, on-line user engagement, and paper citations [17], [18], [19], [20], [11]. Recently, RNN-based point processes such as Recurrent Point process [2], Recurrent Marked Temporal Point process [9] and Neural Hawkes process [21] are broadly discussed. Neural ODEs are also used to model event sequence. Moreover, [22], [23] use a Neural ODEs as decoders of VAEs for event prediction. [24] introduces jumping mechanism into Neural ODEs to model conditional intensity function. However, these existing studies ignore context dependencies of time-series though they are known to influence event occurrence. Moreover, these models mainly focus on the next event type and occurrence timing instead of the long-term forecasting. In this paper, we consider predicting the probability of occurrence for each type of events over the forecasting horizon. In this way, our approach provides richer information about the future for better decision making.

The task of event prediction can also be cast under the framework of time-to-event analysis or survival analysis [25], [26], [27]. Time-to-event analysis is widely used in medical and clinical studies [28], [29], [30], [31], [32], [33], [34], [35]. The main focus of time-to-event analysis is to investigate the distribution of time duration of time until the event of interest happens. There have been studies about jointly modeling time-series with events [36], [37], [38]. A limitation with these methods is that the predicted event only occurs once in the future. In contrast, we consider the problem where events are inter-correlated, and may recurrently occur many times over time.

2.2 Time-Series Forecasting

Traditional statistical forecasting models include linear autoregressive models, such as ARIMA [39], Exponential Smoothing [40] and VAR [41]. They are well-understood and still competitive in many forecasting competitions [4]. Deep neural networks have been proposed to learn from multiple related time-series by fusing traditional models, such as DeepAR [8], RNN and Gaussian copula process model [42], Deep State Space models [7] and its interpretable version [43], Deep Factor models [44].

Another type of network architecture for multi-horizon forecasting involves sequence-to-sequence models [45], [46], which are powerful tools in the domain of Natural Language Processing (NLP), and are often used in machine translation tasks. Rather than modeling each time point within a sequence in an autoregressive manner, the sequence-to-sequence models use an 'encoder-deocoder' framework, to learn a mapping between arbitrarily long sequences through an intermediate encoded state. Sequence-to-sequence models are known to the forecasting community by winning a Kaggle forecasting competition[2]. Its loss function, [47] evaluation metrics [48] and generalization bounds [5] are well-studied in the context of multi-horizon time-series forecasting.

The methods mentioned above are for evenly-sampled synchronous time-series. Models for other types of time-series such as multi-rate time-series [49] and irregularly-sampled time-series [13], [50], [22], [51] are developed, however, they have not been able to generalize to model heterogeneous temporal data.

2. https://www.kaggle.com/c/web-traffic-time-series-forecasting

As shown from the above works, time series forecasting is limited to time series itself, little work has been done for considering the event information, which is the gap to fill by this paper.

3 PRELIMINARIES

3.1 Conditional Variational Autoencoder

Conditional Variational Autoencoder (CVAE) [52] is a conditional directed graphical model which generates output variables $\mathbf{x}$ conditioned on both latent variables $\mathbf{z}$ and condition variables $\mathbf{y}$. In the original Variational Autoencoder (VAE) [53], $\mathbf{x}$ is generated conditioned on only $\mathbf{z}$: $\mathbf{z} \sim p_\theta(\mathbf{z}), \mathbf{x} \sim p_\theta(\mathbf{x}|\mathbf{z})$, in which way the outputs are beyond control. To do conditional generation, $\mathbf{x}$ is generated conditioned on both $\mathbf{z}$ and $\mathbf{y}$ in CVAE: $\mathbf{z} \sim p_\theta(\mathbf{z}), \mathbf{x} \sim p_\theta(\mathbf{x}|\mathbf{z},\mathbf{y})$. The CVAE model is trained based on stochastic gradient variational Bayes (SGVB) [53] framework, using the variational lower bound of the log-likelihood as a surrogate objective function. The variational lower bound can be written as:

$$\log p_\theta(\mathbf{x}|\mathbf{y}) \geq -\mathrm{KL}(q_\phi(\mathbf{z}|\mathbf{x},\mathbf{y}) \| p_\theta(\mathbf{z}|\mathbf{y})) + \mathbb{E}_{q_\phi}[\log p_\theta(\mathbf{x}|\mathbf{y},\mathbf{z})] \tag{1}$$

In this framework, $q_\phi(\mathbf{z}|\mathbf{x},\mathbf{y})$ is the proposal distribution, also known as the recognition model, to approximate the true posterior. $\mathbf{x}$ is generated through the distribution $p_\theta(\mathbf{x}|\mathbf{y},\mathbf{z})$, also known as the generation model. Both the recognition and generation models are implemented using neural networks. Assuming Gaussian latent variables and approximating the second term by drawing samples $\mathbf{z}^{(l)} \sim q_\phi(\mathbf{z}|\mathbf{x},\mathbf{y})$ $(l = 1, \cdots, L)$, the variational lower bound can be further written as:

$$\begin{aligned} &\tilde{\mathcal{L}}_{\text{CVAE}}(\mathbf{x},\mathbf{y};\theta,\phi) \\ = &-\mathrm{KL}(q_\phi(\mathbf{z}|\mathbf{x},\mathbf{y}) \| p_\theta(\mathbf{z}|\mathbf{y})) + \frac{1}{L}\sum_{l=1}^{L} \log p_\theta(\mathbf{x}|\mathbf{y},\mathbf{z}^{(l)}) \end{aligned} \tag{2}$$

Notice that $\mathbf{z}$ is sampled using reparameterization trick: $\mathbf{z}^{(l)} = g_\phi(\mathbf{x},\mathbf{y},\epsilon^{(l)})$, $\epsilon^{(l)} \sim \mathcal{N}(\mathbf{0},\mathbf{I})$, which allows error backpropagation in the training process. Through the methods above, CVAE can be trained efficiently using stochastic gradient descent (SGD).

CVAE framework has been successfully applied to process sequence data in natural language processing [54], [55].

3.2 Neural Ordinary Differential Equations

Neural Ordinary Differential Equations (Neural ODEs) [22] are a family of continuous-depth neural networks. Different from conventional discrete-depth networks (such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs)) which model the transformation between two adjacent layers:

$$\mathbf{h}_{t+1} - \mathbf{h}_t = f_\theta(\mathbf{h}_t) \tag{3}$$

where $t \in \mathbb{Z}_{\geq 0}$ denotes the discrete depth and $\mathbf{h}_t$ is the hidden state of layer t.

Neural ODEs uses ordinary differential equation to model the continuous evolution dynamics of hidden states:

$$\frac{\mathrm{d}\mathbf{h}(t)}{\mathrm{d}t} = f_\theta(\mathbf{h}(t), t) \tag{4}$$

Notations	Descriptions
T, τ	length of observing and predicting time window
$\mathbf{x}_t$	value of time-series measured at time t
c_j, t_j	type and occurrence time of the j-th event
C	number of event types
n, m	number of events in observing and predicting window
$\mathcal{X}_{1:T}, \mathcal{X}_{T+1:T+\tau}$	past and future time-series segmented by time T
$\mathcal{H}_{0^+:T}, \mathcal{H}_{T^+:T+\tau}$	past and future event sequence
$\mathcal{P}, \mathcal{F}$	past and future data, including both time series and events
$\mathcal{Z}$	latent variable of CVAE
θ, ϕ	variables of generative and recognition model in CVAE
φ, ζ, ψ	feature extractors (i.e. functions) modeled by multilayer perceptrons i.e. MLP
$\mathbf{h}_i^{\mathcal{X}}, \mathbf{h}_{t_j}^{\mathcal{H}}$	hidden state of past time-series and event sequence encoder at time i and t_j
$\mathbf{h}_i^{\mathcal{F}\mathcal{X}}, \mathbf{h}_{t_j}^{\mathcal{F}\mathcal{H}}$	hidden state of future time-series and event sequence decoder at time i and t_j

TABLE 1: Notations and descriptions

Here $t \in \mathbb{R}_{\geq 0}$ denotes the continuous depth. f_θ represents the evolution dynamics of hidden states over t, which can be parameterized by a conventional neural network. Neural ODEs can be trained by end-to-end backpropagation algorithm. Once f_θ is well-trained, given the initial value $\mathbf{h}(0)$, hidden state of any depth $\mathbf{h}(t)$ can be easily computed by a black-box differential equation solver:

$$\mathbf{h}(t_0), \ldots, \mathbf{h}(t_N) = \text{ODESolve}(f_\theta, \mathbf{h}(0), t_0, \ldots, t_N) \quad (5)$$

Thanks to its continuous-time (depth) property, Neural ODEs can be used to model irregularly-sampled time-series and temporal point processes [22], [23], [51], [24].

4 Method

This section introduces the proposed method. Section 4.1 introduces our problem formulation. Section 4.2 describes the overall framework of our model. Section 4.3 shows how we model the probability of sequential data, which is of great importance for any CVAE model. Section 4.4 introduces how we implement our model, including structure, training and forecasting processes of our neural networks. The important notations used in this paper are shown in Table 1.

4.1 Problem Formulation

We assume that past time-series and event sequence may interact with each other. Observing the time-series and event sequence during a time interval $(0, T]$, our goal is to predict the future time-series and events during $(T, T+\tau]$, where $T, \tau \in \mathbb{Z}_{>0}$. The problem is illustrated in Figure 1.

More formally, we use $\mathcal{X}_{1:T} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T\}$ to denote an M-dimensional past time-series measured at T consecutive steps $\{1, 2, \ldots T\}$, where each $\mathbf{x}_t \in \mathbb{R}^M$. A potentially correlated event sequence is denoted as $\mathcal{H}_{0^+:T} = \{(c_1, t_1), (c_2, t_2), \ldots, (c_n, t_n)\}$, where $c_i \in \{1, 2, \ldots, C\}$ is the event type of the i-th event and $\{t_i \in \mathbb{R}_{>0} | 0 < t_i \leq T, t_{i-1} < t_i\}$ is the time of occurrence. Note that a time-series is measured at evenly-sampled discrete time points, but events in an event sequence happen irregularly in continuous-time domain.

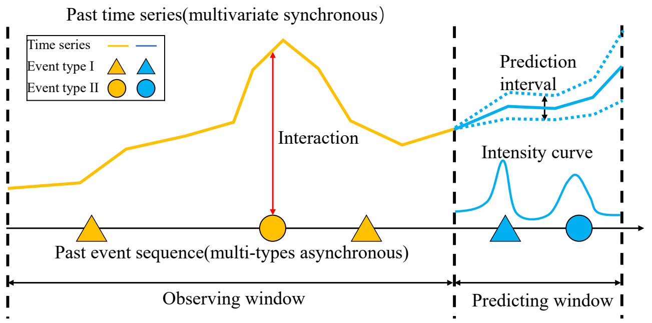


Fig. 1: Illustration of collaborative time-series and event sequence forecasting. Past time-series and event sequence are in orange, predicted time-series and event sequence are in blue. Since past time-series and event sequence interact with each other, a integrated model is needed to disentangle such exogenous factors. Note that our model performs probabilistic prediction instead of deterministic prediction, therefore we get confidence intervals of time-series and conditional intensity function of event sequence.

Given the past time-series $\mathcal{X}_{1:T}$ and event sequence $\mathcal{H}_{0^+:T}$ in observing window $(0, T]$, we want to estimate the probability distribution of future data in predicting window $(T, T+\tau]$ (see Fig. 1):

$$p(\mathcal{X}_{T+1:T+\tau}, \mathcal{H}_{T^+:T+\tau} | \mathcal{X}_{1:T}, \mathcal{H}_{0^+:T}, \Phi) \quad (6)$$

Here $\mathcal{X}_{T+1:T+\tau} = \{\mathbf{x}_{T+1}, \mathbf{x}_{T+2}, \ldots, \mathbf{x}_{T+\tau}\}$ denotes the future time-series in the predicting window $(T, T+\tau]$. $\mathcal{H}_{T^+:T+\tau} = \{(c_{n+1}, t_{n+1}), (c_{n+2}, t_{n+2}), \ldots, (c_{n+m}, t_{n+m})\}$ denotes the future event sequence in this window. Φ denotes the model parameters. $n, m \in \mathbb{Z}_{\geq 0}$ are the number of events in observing and predicting window, and vary in different sequences. $n = 0$ or $m = 0$ means that there is no event in the corresponding window. Note that the observing window $(0, T]$ is actually $\{1, 2, \ldots, T\}$ for time-series, same for predicting window.

4.2 Generative Forecasting Framework

We use $\mathcal{P} = \{\mathcal{X}_{1:T}, \mathcal{H}_{0^+:T}\}$ to denote past time-series and event sequence, $\mathcal{F} = \{\mathcal{X}_{T+1:T+\tau}, \mathcal{H}_{T^+:T+\tau}\}$ to denote future data. In general, we want to estimate the probability distribution of future conditioned on the past, denoted as $p_\theta(\mathcal{F}|\mathcal{P})$, where θ is parameters for generative model. To do so, we introduce a latent variable $\mathcal{Z}$ and use the CVAE framework [52]. Overall, the latent variable $\mathcal{Z}$ is generated from prior distribution $p_\theta(\mathcal{Z}|\mathcal{P})$ and future data $\mathcal{F}$ is generated from the generative model $p_\theta(\mathcal{F}|\mathcal{P}, \mathcal{Z})$: $\mathcal{Z} \sim p_\theta(\mathcal{Z}|\mathcal{P}), \mathcal{F} \sim p_\theta(\mathcal{F}|\mathcal{P}, \mathcal{Z})$. Marginalizing out $\mathcal{Z}$, we can obtain the target distribution:

$$\begin{aligned} p_\theta(\mathcal{F}|\mathcal{P}) &= \int_{\mathcal{Z}} p_\theta(\mathcal{F}, \mathcal{Z}|\mathcal{P}) \, d\mathcal{Z} \\ &= \int_{\mathcal{Z}} p_\theta(\mathcal{F}|\mathcal{P}, \mathcal{Z}) p_\theta(\mathcal{Z}|\mathcal{P}) \, d\mathcal{Z} \end{aligned} \quad (7)$$

Due to the intractable posterior distribution, it is hard to optimize the log-likelihood directly. Instead, the CVAE is trained by optimizing the variational lower bound of log-likelihood, i.e.:

$$\begin{aligned}
&\log p_\theta(\mathcal{F}|\mathcal{P}) \\
=&\log \int_{\mathcal{Z}} q_\phi(\mathcal{Z}|\mathcal{P},\mathcal{F}) \frac{p_\theta(\mathcal{F},\mathcal{Z}|\mathcal{P})}{q_\phi(\mathcal{Z}|\mathcal{P},\mathcal{F})} \mathrm{d}\mathcal{Z} \\
\geq& \int_{\mathcal{Z}} q_\phi(\mathcal{Z}|\mathcal{P},\mathcal{F}) \log \frac{p_\theta(\mathcal{F},\mathcal{Z}|\mathcal{P})}{q_\phi(\mathcal{Z}|\mathcal{P},\mathcal{F})} \mathrm{d}\mathcal{Z} \\
=& \int_{\mathcal{Z}} q_\phi(\mathcal{Z}|\mathcal{P},\mathcal{F}) \log \frac{p_\theta(\mathcal{Z}|\mathcal{P}) p_\theta(\mathcal{F}|\mathcal{Z},\mathcal{P})}{q_\phi(\mathcal{Z}|\mathcal{P},\mathcal{F})} \mathrm{d}\mathcal{Z} \\
=& -\mathrm{KL}(q_\phi(\mathcal{Z}|\mathcal{P},\mathcal{F}) \| p_\theta(\mathcal{Z}|\mathcal{P})) + \mathbb{E}_{q_\phi}[\log p_\theta(\mathcal{F}|\mathcal{P},\mathcal{Z})] \\
=& \mathcal{L}(\mathcal{P},\mathcal{F};\theta,\phi)
\end{aligned} \tag{8}$$

where KL is Kullback-Leibler (KL) divergence. $q_\phi(\mathcal{Z}|\mathcal{P},\mathcal{F})$ is the "recognition" model introduced to approximate the intractable true posterior distribution $p_\theta(\mathcal{Z}|\mathcal{P},\mathcal{F})$. It is also known as "encoder" and we use q_ϕ for simplification. $p_\theta(\mathcal{F}|\mathcal{P},\mathcal{Z})$ is the "generative" model used to generate future distribution from latent variable and past data. "generative" model is also known as "decoder". $p_\theta(\mathcal{Z}|\mathcal{P})$ is the prior distribution, which is often modeled by a neural network in CVAE.

Assuming $q_\phi(\mathcal{Z}|\mathcal{P},\mathcal{F})$ and $p_\theta(\mathcal{Z}|\mathcal{P})$ to be Gaussian distributions, the KL term in Eq. 8 is differentiable as it has an analytical solution, while the expectation term is not. Hence we draw $K \in \mathbb{Z}_{>0}$ samples from the "recognition" model to approximate the expectation term and use the reparameterization trick [56] to make the sample operation differentiable. The empirical bound can be written as:

$$\begin{aligned}
&\tilde{\mathcal{L}}_{\text{CVAE}}(\mathcal{P},\mathcal{F};\theta,\phi) \\
=& -\mathrm{KL}\left(q_\phi(\mathcal{Z}|\mathcal{P},\mathcal{F}) \| p_\theta(\mathcal{Z}|\mathcal{P})\right) + \frac{1}{K}\sum_{k=1}^{K} \log p_\theta(\mathcal{F}|\mathcal{P},\mathcal{Z}^{(k)})
\end{aligned} \tag{9}$$

where $\mathcal{Z}^{(k)} = g_\phi(\mathcal{P},\mathcal{F},\boldsymbol{\epsilon}^{(k)}), \boldsymbol{\epsilon}^{(k)} \sim \mathcal{N}(\mathbf{0},\mathbf{I})$. g_ϕ is a deterministic differentiable function to perform sample operation. Therefore, we can compute the empirical lower bound and backpropagate it to perform training.

Remarks. Our approach

4.3 Conditional Probability Modeling

It shall be noted that $q_\phi(\mathcal{Z}|\mathcal{P},\mathcal{F})$ and $p_\theta(\mathcal{Z}|\mathcal{P})$ in Eq. 8 can be easily modeled with a simple assumption of $\mathcal{Z}$, such as multivariate Gaussian, while $p_\theta(\mathcal{F}|\mathcal{P},\mathcal{Z})$ can not. The main challenge is the sequential structure of $\mathcal{F}$.

We assume that time-series and event sequence are conditionally independent given $\mathcal{P},\mathcal{Z}$, i.e.:

$$\begin{aligned}
p_\theta(\mathcal{F}|\mathcal{P},\mathcal{Z}) &= p_\theta(\mathcal{X}_{T+1:T+\tau},\mathcal{H}_{T^+:T+\tau}|\mathcal{P},\mathcal{Z}) \\
&= \underbrace{p_\theta^{\mathcal{X}}(\mathcal{X}_{T+1:T+\tau}|\mathcal{P},\mathcal{Z})}_{\text{probability of time-series}} \underbrace{p_\theta^{\mathcal{H}}(\mathcal{H}_{T^+:T+\tau}|\mathcal{P},\mathcal{Z})}_{\text{probability of event sequence}}
\end{aligned} \tag{10}$$

Here $p_\theta^{\mathcal{X}}$ denotes the probability distribution of future time-series and $p_\theta^{\mathcal{H}}$ denotes the probability distribution of future event sequence. θ denotes the generative model, we omit it in this section for better readability. Next, we will show how to model time series $p^{\mathcal{X}}(\mathcal{X}_{T+1:T+\tau}|\mathcal{P},\mathcal{Z})$ and event sequence $p^{\mathcal{H}}(\mathcal{H}_{T^+:T+\tau}|\mathcal{P},\mathcal{Z})$ respectively.

4.3.1 *Probability of time-series.*

Assuming each step depends on $\mathcal{P},\mathcal{Z}$ and earlier predicted steps, the probability distribution of time-series can be computed by:

$$\begin{aligned}
&p^{\mathcal{X}}(\mathcal{X}_{T+1:T+\tau}|\mathcal{P},\mathcal{Z}) \\
=&p^{\mathcal{X}}(\mathbf{x}_{T+1}|\mathcal{P},\mathcal{Z}) p^{\mathcal{X}}(\mathbf{x}_{T+2}|\mathcal{P},\mathcal{Z},\tilde{\mathcal{X}}_{T+1:T+1}) \ldots \\
&p^{\mathcal{X}}(\mathbf{x}_{T+\tau}|\mathcal{P},\mathcal{Z},\tilde{\mathcal{X}}_{T+1:T+\tau-1}) \\
=&\prod_{i=1}^{\tau} p^{\mathcal{X}}(\mathbf{x}_{T+i}|\mathcal{P},\mathcal{Z},\tilde{\mathcal{X}}_{T+1:T+i-1})
\end{aligned} \tag{11}$$

where $\tilde{\mathcal{X}}$ denotes predicted values. We let $\tilde{\mathcal{X}}_{T+1:T+i-1} = \mathbf{x}_{T-1}$ when $i=1$ for practical computing.

4.3.2 *Probability of event sequence.*

Different from time-series measured at discrete grid time points, events happen irregularly in continuous-time domain and the number of events in the future varies from sequence to sequence. Therefore, it is hard to model the probability distribution of an event sequence directly. We resort to conditional intensity function to model the probability of event sequence.

A multivariate event sequence with C types of events can be viewed as the combination of C univariate sequences. Each univariate sequence is formulated by events with the same type label. Formally:

$$\begin{aligned}
&\mathcal{H}_{T^+:T+\tau} \\
=&\{(c_{n+1},t_{n+1}),(c_{n+2},t_{n+2}),\ldots(c_{n+m},t_{n+m})\} \\
=&\cup_{c=1}^{C}\ \{(c,t_1^c),(c,t_2^c),\ldots(c,t_{m_c}^c)\} \\
=&\cup_{c=1}^{C}\ \mathcal{H}_{T^+:T+\tau}^c
\end{aligned} \tag{12}$$

Here, t_j^c denotes occurrence time of the j-th event of type c. m_c is the number of events of type c with $\sum_{c=1}^{C} m_c = m$. $\mathcal{H}_{T^+:T+\tau}^c$ denotes the event sequence of type c in the future.

With the conditional independence assumption, the probability of a multi-type event sequence can be modeled by:

$$p^{\mathcal{H}}(\mathcal{H}_{T^+:T+\tau}|\mathcal{P},\mathcal{Z}) = \prod_{c=1}^{C} p^{\mathcal{H}}(\mathcal{H}_{T^+:T+\tau}^c|\mathcal{P},\mathcal{Z}) \tag{13}$$

Given the intensity function $\lambda^c(t|\mathcal{P},\mathcal{Z})$, the probability of a univariate event sequence $\mathcal{H}_{T^+:T+\tau}^c$ can be modeled as:

$$\begin{aligned}
&p^{\mathcal{H}}(\mathcal{H}_{T^+:T+\tau}^c|\mathcal{P},\mathcal{Z}) \\
=&p^{\mathcal{H}}(t_1^c,t_2^c,\ldots,t_{m_c}^c|\mathcal{P},\mathcal{Z}) \\
=&\prod_{j=1}^{m_c} \lambda^c(t_j^c|\mathcal{P},\mathcal{Z}) e^{-\int_T^{T+\tau} \lambda^c(t|\mathcal{P},\mathcal{Z})\mathrm{d}t}
\end{aligned} \tag{14}$$

As the event type in each $\mathcal{H}_{T^+:T+\tau}^c$ is the same, we can get rid of event type and only need to model the intensity function for each type of event.

4.4 Implementations

The architecture of our model is shown in Fig. 2. We use the CVAE framework: past time-series and event sequence are input into the prior network for feature extraction and latent distribution computation. Then the feature and sampled latent variable is input into decoders for future prediction. We know describe components in our model in detail.

4.4.1 Prior network and recognition model

We use prior network to compute the prior distribution $p_\theta(\mathcal{Z}|\mathcal{P})$ and recognition Model (also known as encoder) to compute $q_\phi(\mathcal{Z}|\mathcal{P},\mathcal{F})$ to approximate true posterior distribution.

Time-Aware Hybrid RNN. As we have discussed above, time-series is measured at discrete grid time points and events happen irregularly in continuous-time domain. This difference makes it challenging to model these two kinds of data jointly, especially for time order information preservation, e.g., an event can only be influenced by time-series earlier than it. We use Time-Aware Hybrid RNN to extract feature from past time-series and event sequence jointly.

Time-Aware Hybrid RNN is consist of two RNNs: a Time-series RNN and an Event RNN. The time-series transits at grid time points $1, 2, \ldots, T$ where values are measured. The Event RNN, following [9], takes event type and time duration as input and transits when events occur. To jointly model them and preserve time order, we also take temporal features which are uncorrelated to time-series and events as input for both RNNs. Assuming we have a time-series with length T and a event sequence with length n, Time-Aware Hybrid RNN performs the following operations:

$$\begin{aligned} \mathbf{h}_i^{\mathcal{X}} &= f([\varphi(\mathbf{x}_i), \psi(\mathbf{t}_i)], \mathbf{h}_{i-1}^{\mathcal{X}}) && \text{for } i = 1, \ldots, T \\ \mathbf{h}_{t_j}^{\mathcal{H}} &= g([\zeta(c_j), \Delta t_j, \psi(\mathbf{t}_j)], \mathbf{h}_{t_{j-1}}^{\mathcal{H}}) && \text{for } j = 1, \ldots, n \end{aligned} \tag{15}$$

where f and g denote time-series RNN and Event RNN, $\Delta t_j = t_j - t_{j-1}$ denotes the time duration between two adjacent events. φ and ζ extract features from time-series and event type respectively. The blod $\mathbf{t_i}$ denotes temporal features (such as absolute timestamp, hour of day and day of week) and we use ψ to embed them into a feature vector. $[\cdot,\cdot]$ denotes concatenation operation. In our experiments, we use MLPs for φ, ζ and ψ. f and g are parameterized by GRUs and initialized with zero vectors for $i = 0$ and $j = 0$.

Auxiliary Transition Unit. When modeling very long sequences, a typical practice is to split time-series into chunks that are overlapped at the time axis [8]. However, with the introduction of event sequences, such an approach causes a problem. Because events occur irregularly, many adjacent chunks may share the same set of events, where the event type, timing, and all other features are the same. However, the forecast targets can be different for the adjacent chunks, which means that the same event sequence input may reflect different targets, making it difficult for the model to capture features within the event sequence. To solve the problem, we introduce Auxiliary Transition Unit, whose basic idea is to let the asynchronous RNN know the exact end time as the time-series RNN, which is done by an auxiliary transit at T with zero event type input:

$$\mathbf{h}_T^{\mathcal{H}} = g\left([\zeta(0), \triangle t_T, \psi(\mathbf{t}_T)], \mathbf{h}_{t_n}^{\mathcal{H}}\right) \tag{16}$$

where $\triangle t_T = T - t_n$. By doing so, our model can be trained in batch effortlessly, without worrying about data conflicts.

Synergetic Layer. We use $\mathbf{h}_T^{\mathcal{X}}$ and $\mathbf{h}_T^{\mathcal{H}}$ to represent features extracted from past time-series and event sequence by Time-Aware Hybrid RNN. Synergetic Layer transfers these features into prior distribution $p_\theta(\mathcal{Z}|\mathcal{P})$ and approximate posterior distribution $q_\phi(\mathcal{Z}|\mathcal{P},\mathcal{F})$. Following [56], we assume these two distributions are multivariate Gaussians with diagonal covariance. For the prior distribution, we have:

$$\begin{aligned} p_\theta(\mathcal{Z}|\mathcal{P}) &= \mathcal{N}(\boldsymbol{\mu}_\theta, \boldsymbol{\Sigma}_\theta) \\ \boldsymbol{\mu}_\theta &= \mathrm{MLP}_\theta^{\mu}([\mathbf{h}_T^{\mathcal{X}}, \mathbf{h}_T^{\mathcal{H}}]) \\ \boldsymbol{\Sigma}_\theta &= \mathrm{diag}^2(\mathrm{MLP}_\theta^{\Sigma}([\mathbf{h}_T^{\mathcal{X}}, \mathbf{h}_T^{\mathcal{H}}])) \end{aligned} \tag{17}$$

As for the recognition model, we input past and future data $\mathcal{X}_{1:T+\tau}$ and $\mathcal{H}_{0^+:T+\tau}$ into the same Time-Aware Hybrid RNN to get extracted features $\mathbf{h}_{T+\tau}^{\mathcal{X}}$ and $\mathbf{h}_{T+\tau}^{\mathcal{H}}$. Then another pair of MLPs are utilized to compute the parameters of $q_\phi(\mathcal{Z}|\mathcal{P},\mathcal{F})$:

$$\begin{aligned} q_\phi(\mathcal{Z}|\mathcal{P},\mathcal{F}) &= \mathcal{N}(\boldsymbol{\mu}_\Phi, \boldsymbol{\Sigma}_\Phi) \\ \boldsymbol{\mu}_\Phi &= \mathrm{MLP}_\Phi^{\mu}([\mathbf{h}_{T+\tau}^{\mathcal{X}}, \mathbf{h}_{T+\tau}^{\mathcal{H}}]) \\ \boldsymbol{\Sigma}_\Phi &= \mathrm{diag}^2(\mathrm{MLP}_\Phi^{\Sigma}([\mathbf{h}_{T+\tau}^{\mathcal{X}}, \mathbf{h}_{T+\tau}^{\mathcal{H}}])) \end{aligned} \tag{18}$$

4.4.2 Generative model

Given the extracted past features $\mathbf{h}_T^{\mathcal{X}}, \mathbf{h}_T^{\mathcal{H}}$ and sampled $\mathcal{Z} \sim q_\phi(\mathcal{Z}|\mathcal{P},\mathcal{F})$, the generative model parameterizes the future distribution $p_\theta(\mathcal{F}|\mathcal{P},\mathcal{Z})$. More specifically, the time-series decoder models future time-series distribution $p_\theta^{\mathcal{X}}(\mathcal{X}_{T+1:T+\tau}|\mathcal{P},\mathcal{Z})$ and the event decoder models future event sequence distribution $p_\theta^{\mathcal{H}}(\mathcal{H}_{T^+:T+\tau}|\mathcal{P},\mathcal{Z})$.

Auto-regressive Time-series Decoder. We use an RNN to model the hidden state at each future step and then map the hidden state to some probability distribution. Formally, the hidden state is initialized as:

$$\mathbf{h}_T^{\mathcal{F}\mathcal{X}} = \mathrm{MLP}_\theta^{\mathcal{X}}([\mathbf{h}_T^{\mathcal{X}}, \mathbf{h}_T^{\mathcal{H}}, \mathrm{MLP}_\theta^{\mathcal{Z}}(\mathcal{Z})]) \tag{19}$$

where $\mathbf{h}_T^{\mathcal{F}\mathcal{X}}$ denotes the hidden state of future time-series and should be distinguished from $\mathbf{h}_T^{\mathcal{X}}$. Then we iteratively compute the probability distribution at each time step $i = T+1, \ldots, T+\tau$ as:

$$\begin{aligned} \mathbf{h}_i^{\mathcal{F}\mathcal{X}} &= d_\theta([\varphi(\mathbf{x}_{i-1}), \psi(\mathbf{t_i})], \mathbf{h}_{i-1}^{\mathcal{F}\mathcal{X}}) \\ \mathbf{x}_i &\sim \mathcal{PD}(R_\theta(\mathbf{h}_i^{\mathcal{F}\mathcal{X}})) \end{aligned} \tag{20}$$

where d_θ is an RNN, φ and ψ are same feature extractors we use in Eq. 15. Here we do not set the specific form of the distribution. Instead, we assume it follows an abstract distribution $\mathcal{PD}$ with parameters R. $\mathcal{PD}$ can take different forms for different kinds of data, making our model more flexible. Note that when $i = T+1$, $\mathbf{x}_{i-1}$ is not sampled from distribution but the true value from past series.

Event Decoder. Because the number of events in the predicting window $(T, T+\tau]$ is unfixed, it is hard to model the probability of future event sequence directly. Instead, we model the conditional intensity function and use Eq. 13 and Eq. 14 to compute $p_\theta^{\mathcal{H}}(\mathcal{H}_{T^+:T+\tau}|\mathcal{P},\mathcal{Z})$.

Here we use neural ODEs to model the intensity function, which enables to model the complex dynamics of intensity function without assuming its functional form like [9]. Formally, we use the same technique as Eq. 19 to initialize

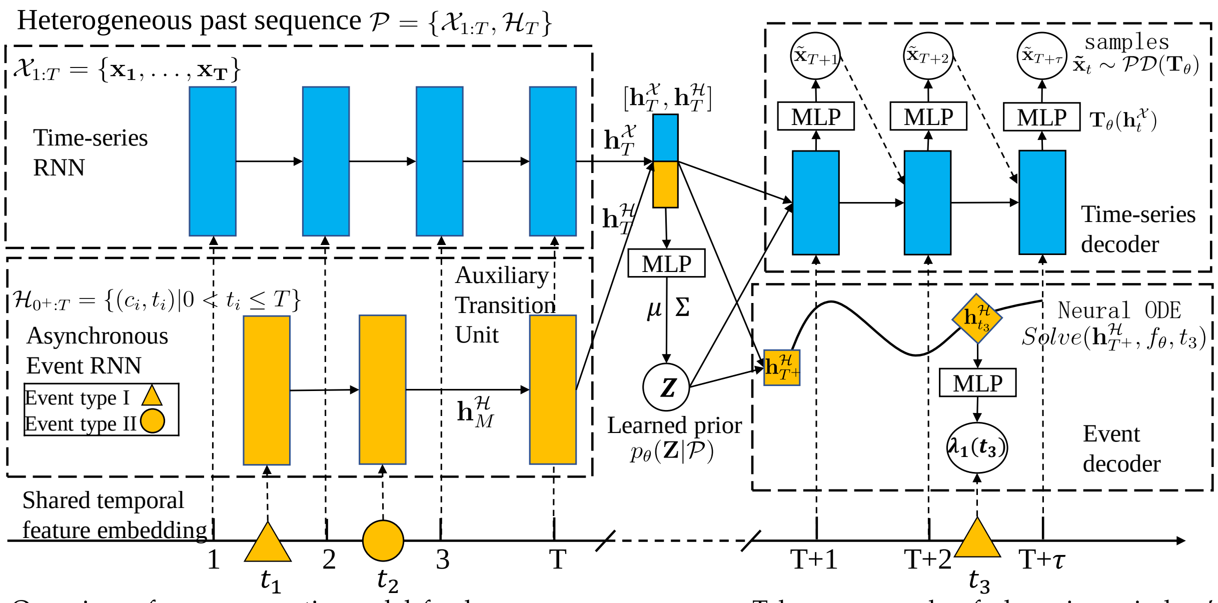

Heterogeneous past sequence $\mathcal{P} = \{\mathcal{X}_{1:T}, \mathcal{H}_T\}$
$\mathcal{X}_{1:T} = \{\mathbf{x_1}, \dots, \mathbf{x_T}\}$
Time-series RNN
$[\mathbf{h}_T^{\mathcal{X}}, \mathbf{h}_T^{\mathcal{H}}]$
$\mathbf{h}_T^{\mathcal{X}}$
$\mathbf{h}_T^{\mathcal{H}}$
MLP
μ Σ
$\mathbf{Z}$
Learned prior $p_\theta(\mathbf{Z}|\mathcal{P})$
$\mathcal{H}_{0^+:T} = \{(c_i, t_i)|0 < t_i \leq T\}$
Asynchronous Event RNN
Event type I ▲
Event type II ●
Auxiliary Transition Unit
$\mathbf{h}_M^{\mathcal{H}}$
Shared temporal feature embedding
1 2 3 T T+1 T+2 T+τ
t_1 t_2 t_3
$\tilde{\mathbf{x}}_{T+1}$ $\tilde{\mathbf{x}}_{T+2}$ $\tilde{\mathbf{x}}_{T+\tau}$
samples $\tilde{\mathbf{x}}_t \sim \mathcal{PD}(\mathbf{T}_\theta)$
MLP MLP MLP $\mathbf{T}_\theta(\mathbf{h}_t^{\mathcal{X}})$
Time-series decoder
$\mathbf{h}_{T^+}^{\mathcal{H}}$
$\mathbf{h}_{t_3}^{\mathcal{H}}$
Neural ODE $Solve(\mathbf{h}_{T^+}^{\mathcal{H}}, f_\theta, t_3)$
MLP
$\lambda_1(t_3)$
Event decoder

Fig. 2: Overview of our synergetic model for homogeneous sequence. Take an example of observing window's length $T = 4$, and $n = 2$ events occur within the time range. The length of predicting window $\tau = 3$, and $m = 1$ event occurs within the time range. In training process, past data $\mathcal{P}$ is input into Time-series RNN and Asynchronous Event RNN to compute the prior distribution $p_\theta(\mathcal{Z}|\mathcal{P})$; past and future data $(\mathcal{P}, \mathcal{F})$ is input into same RNNs to get $q_\phi(\mathcal{Z}|\mathcal{P}, \mathcal{F})$ to approximate the posterior distribution (not shown in the plot). Time-series Decoder outputs the distribution of x_t at each step to compute $p_\theta^{\mathcal{X}}(\mathcal{X}_{T+1:T+\tau}|\mathcal{P}, \mathcal{Z})$. Event Decoder outputs λ at points when events occur in real to compute $p_\theta^{\mathcal{H}}(\mathcal{H}_{T^+:T+\tau}|\mathcal{P}, \mathcal{Z})$. The model is learned to maximize the conditional probability $p(\mathcal{F}|\mathcal{P})$ by minimizing $\mathrm{KL}(q_\phi(\mathcal{Z}|\mathcal{P}, \mathcal{F})\|p_\theta(\mathcal{Z}|\mathcal{P})) - \mathbb{E}_{q_\phi}\left[\log(p_\theta^{\mathcal{X}}(\mathcal{X}_{T+1:T+\tau}|\mathcal{P}, \mathcal{Z})p_\theta^{\mathcal{H}}(\mathcal{H}_{T^+:T+\tau}|\mathcal{P}, \mathcal{Z}))\right]$. In predicting process, Event Decoder outputs λ at evenly distributed grid points to get intensity curves for further prediction.

the hidden state of Event Decoder:

$$\mathbf{h}_{T^+}^{\mathcal{FH}} = \mathrm{MLP}_\theta^{\mathcal{H}}([\mathbf{h}_T^{\mathcal{X}}, \mathbf{h}_T^{\mathcal{H}}, \mathrm{MLP}_\theta^{\mathcal{Z}}(\mathcal{Z})]) \tag{21}$$

where $\mathrm{MLP}_\theta^{\mathcal{H}}$ denotes the MLP for event sequence, different from the one in Eq. 19. Then the hidden state at time points $t_{n+1}, t_{n+2}, \dots, t_{n+m}$ can be computed by:

$$\begin{aligned}&\mathbf{h}_{t_{n+1}}^{\mathcal{FH}}, \mathbf{h}_{t_{n+2}}^{\mathcal{FH}}, \dots, \mathbf{h}_{tn+m}^{\mathcal{FH}} \\ =&\mathrm{ODESolve}(\mathbf{h}_{T^+}^{\mathcal{FH}}, I_\theta, t_{n+1}, t_{n+2}, \dots, t_{n+m})\end{aligned} \tag{22}$$

where $t_{n+1}, t_{n+2}, \dots, t_{n+m}$ are time points when real event occur and should be known in training process. I_θ is a function to model the changing dynamics of hidden state: $I_\theta(\mathbf{h}_t^{\mathcal{FH}}, t) = \frac{\mathbf{h}_t^{\mathcal{FH}}}{\mathrm{d}t}$. Then an MLP is used to compute the conditional intensity for each type of event at these time points:

$$\boldsymbol{\lambda}(t_{n+j}|\mathcal{P}, \mathcal{Z}) = \mathrm{MLP}_\theta^{\lambda}(\mathbf{h}_{t_{n+j}}^{\mathcal{FH}}) \tag{23}$$

where $\boldsymbol{\lambda}(t_{n+j}|\mathcal{P}, \mathcal{Z}) \in \mathbb{R}^C$, the c-th dimension represents the intensity of type c at that time point. $\mathrm{MLP}_\theta^{\lambda}$ is a network with C output units and SoftPlus function as activation. Moreover, $\int_T^{T+\tau} \lambda^c(t|\mathcal{P}, \mathcal{Z})\mathrm{d}t$ in Eq. 14 can also be computed using neural ODEs by assigning initial state as 0 and $I_\theta = \lambda^c(t|\mathcal{P}, \mathcal{Z})$.

4.4.3 *Training*

Our model is trained by gradient descent to maximize the evidence lower bound (ELBO), as computed in Algorithm 1.

Algorithm 1 computes the ELBO of a single sequence $\{\mathcal{P}, \mathcal{F}\}$. In practice, we use mini batch gradient descent to train the model. Using tricks such as padding and dummy code, we compute the mean ELBO of a batch of sequences and optimize it by gradient descent.

Algorithm 1: ELBO computation for RNN-ODE

```
Input: Past time-series and event sequence
       $\mathcal{P} = \{\mathcal{X}_{1:T}, \mathcal{H}_{0^+:T}\}$; Future time-series and
       event sequence $\mathcal{F} = \{\mathcal{X}_{T+1:T+\tau}, \mathcal{H}_{T^+:T+\tau}\}$;
       Number of samples to approximate
       expectation $K$;
Output: ELBO of the input data ELBO$(\mathcal{P}, \mathcal{F}; \theta, \phi)$
Compute $p_\theta(\mathcal{Z}|\mathcal{P})$ using Equations 15, 16 and 17;
Compute $q_\phi(\mathcal{Z}|\mathcal{P}, \mathcal{F})$ using Equations 15, 16 and 18;
for $k = 1$ to $K$ do
    Sample $\mathcal{Z}^{(k)} \sim q_\phi(\mathcal{Z}|\mathcal{P}, \mathcal{F})$ using
      reparameterization trick;
    Compute $p_\theta^{\mathcal{X}}(\mathcal{X}_{T+1:T+\tau}|\mathcal{P}, \mathcal{Z}^{(k)})$ using
      Equations 19, 20 and 11;
    Compute $p_\theta^{\mathcal{H}}(\mathcal{H}_{T^+:T+\tau}|\mathcal{P}, \mathcal{Z}^{(k)})$ using
      Equations 21, 22, 23, 14 and 13;
end
ELBO$(\mathcal{P}, \mathcal{F}; \theta, \phi) =$
        $-\mathrm{KL}(q_\phi(\mathcal{Z}|\mathcal{P}, \mathcal{F})\|p_\theta(\mathcal{Z}|\mathcal{P}))$
        $+\frac{1}{K}\sum_{k=1}^{K} \log p_\theta^{\mathcal{X}}(\mathcal{X}_{T+1:T+\tau}|\mathcal{P}, \mathcal{Z}^{(k)})$
        $+\frac{1}{K}\sum_{k=1}^{K} \log p_\theta^{\mathcal{H}}(\mathcal{H}_{T^+:T+\tau}|\mathcal{P}, \mathcal{Z}^{(k)})$
Return ELBO$(\mathcal{P}, \mathcal{F}; \theta, \phi)$;
```

Algorithm 2: Probabilistic Forecasting by Monte-Carlo Sampling

```
Input: Heterogeneous past data $\mathcal{P} = \{\mathcal{X}_{1:T}, \mathcal{H}_{0^+:T}\}$;
       Trained model $p_\theta(\mathcal{F}|\mathcal{P}, \mathcal{Z})$ and $p_\theta(\mathcal{Z}|\mathcal{P})$;
       Forecast horizon $\tau$; number of samples $N$ and
       number of time points in intensity curve $M$;
Output: Mean and confidence intervals for future
        time series; Conditional intensity curves for
        future event sequence;
Compute $\mathbf{h}_T^{\mathcal{X}}$ and $\mathbf{h}_T^{\mathcal{H}}$ by Eq. 15;
Compute $p_\theta(\mathcal{Z}|\mathcal{P})$ by Eq. 17;
for $n = 1$ to $N$ do
    Sample $\mathcal{Z} \sim p_\theta(\mathcal{Z}|\mathcal{P})$;
    $\mathbf{h}_T^{\mathcal{FX}} = \mathrm{MLP}_\theta^{\mathcal{X}}([\mathbf{h}_T^{\mathcal{X}}, \mathbf{h}_T^{\mathcal{H}}, \mathrm{MLP}_\theta^{\mathcal{Z}}(\mathcal{Z})])$;
    for $i = T + 1$ to $T + \tau$ do
        $\mathbf{h}_i^{\mathcal{FX}} = d_\theta([\varphi(\mathbf{x}_{i-1}), \psi(\mathbf{t}_i)], \mathbf{h}_{i-1}^{\mathcal{FX}})$
        $\mathbf{x}_i^{(n)} \sim \mathcal{PD}(R_\theta(\mathbf{h}_i^{\mathcal{FX}}))$
    end
    $\mathbf{h}_{T^+}^{\mathcal{FH}} = \mathrm{MLP}_\theta^{\mathcal{H}}([\mathbf{h}_T^{\mathcal{X}}, \mathbf{h}_T^{\mathcal{H}}, \mathrm{MLP}_\theta^{\mathcal{Z}}(\mathcal{Z})])$
    $\mathbf{h}_{T+\frac{1}{M}\tau}^{\mathcal{FH}}, \mathbf{h}_{T+\frac{2}{M}\tau}^{\mathcal{FH}}, \ldots, \mathbf{h}_{T+\tau}^{\mathcal{FH}}$
    $= \mathrm{ODESolve}(\mathbf{h}_{T^+}^{\mathcal{FH}}, I_\theta, T+\frac{1}{M}\tau, T+\frac{2}{M}\tau, \ldots, T+\tau)$
    for $i = 1$ to $M$ do
        $\boldsymbol{\lambda}^{(n)}(T + \frac{i}{M}\tau|\mathcal{P}, \mathcal{Z}) = \mathrm{MLP}_\theta^{\lambda}(\mathbf{h}_{T+\frac{i}{M}\tau}^{\mathcal{FH}})$
    end
end
Return mean and quantiles of $\mathbf{x}_i^{(n)}$ and
  $\boldsymbol{\lambda}^{(n)}(T + \frac{i}{M}\tau|\mathcal{P}, \mathcal{Z})$ along axis $n$;
```

Algorithm 3: Deterministic Event Prediction from Intensity Curve by Sampling Clustering

```
Input: Univariate intensity curve of event type $c$
       $\lambda_{T^+:T+\tau}^{c}(t|\mathcal{P}, \mathcal{Z}) \in R^{M \times 1}$; Number of
       samples $N$; Minimum weight for a cluster $\epsilon$;
Output: Occurrence time points of events
        corresponding to the intensity curve
Set the pool of time points as empty $S = \{\}$;
for $i = 1$ to $N$ do
    Use thinning algorithm to sample event sequence
      from intensity
      $seq_i = \{t_1, t_2, \ldots, t_{n_i}\} \sim \lambda_{T^+:T+\tau}^{c}(t|\mathcal{P}, \mathcal{Z})$;
    Append sampled time points to the pool
      $S = S \cup seq_i$;
end
Use GMM to cluster time points in $S$ and BIC for
  cluster number selection, denote the selected model
  as $G$ with $k$ clusters;
$events = \{\}$
for $i = 1$ to $k$ do
    if $G_{weights}[i] \geq \epsilon$ then
        $events = events \cup G_{centers}[i]$
end
Return selected cluster centers $events$;
```

4.4.4 *Forecasting*

Probabilistic forecasting of time-series and event sequence. Once our model is well-trained, given past data, we can forecast the future time-series and event sequence for new coming data. Due to the randomness and unobserved external effect, the future can be uncertain, i.e., the same past data leads to different future. This is very common in time-series and event sequence forecasting. Therefore, we perform probabilistic forecasting to provide prediction with uncertainty for decision making. The forecasting approach is shown in Algorithm 2.

Deterministic forecasting of event sequence. Although intensity curves can completely characterize the future event sequence, they are abstract for human but hard to be compared with other deterministic methods. Thus, we need to gain deterministic events from intensity curves. This is done by Algorithm 3.

We predict future events for each type respectively by Algorithm 3, attach their type labels with them, pool them together and sort according to the occurrence time.

5 EXPERIMENTS

We conduct experiments to test the performance of our model. Section 5.1 introduces our experiment settings, including data descriptions and implementation details. Section 5.2 presents results and analysis of time-series forecasting compared with baseline methods. Section 5.3 presents results and analysis of event sequence forecasting. We visualize some of our forecasting results in Section 5.4.

5.1 Protocols

5.1.1 *Dataset description*

Two univariate datasets and one multivariate dataset are used for evaluation, whereby four temporal features are extracted: absolute time, hour-of-day, day-of-week, and month-of-year for hourly-sampled time-series data. The last four weeks are hold-out for test, and the rest for training. We train the model to predict the future day (24 data points) given the past week (24 × 7 = 168 points), along with events within that week.

Electricity[3]. The UCI household electricity dataset contains time-series of electricity usage in kW recorded hourly for 370 clients. A univariate series of length 21,044 is used here, and the time points with fluctuations larger than 30 are extracted as events. The event types are divided into up and down according to the direction.

Traffic[4]. The dataset corresponds to hourly-sampled road occupancy rates in percentiles (0-100%) collected from the California Department of Transportation. The first univariate series of length 17,544 is used for experiment. We extract time points with fluctuations higher than 10% as events, whose types are up and down according to the direction.

Environment. It is a public air quality multivariate dataset [43], which may showcase how atmospheric variables interact with weather events. Four hourly-sampled variables: PM2.5, dew point, temperature and pressure are used, and three types of events are extracted from the minute-level data sources: wind start, wind stop and rain start.

3. https://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams 20112014
4. https://archive.ics.uci.edu/ml/datasets/PEMS-SF

5.1.2 Implementation details

5.1.3 Parameter settings

5.2 Forecasting of time-series

5.2.1 Evaluation metrics

The quality of the forecasting is generally measured using a metric calculated between the predicted and actual values in the forecast horizon. Probabilistic forecasts estimate a probability distribution rather than predicting a single value as point forecasts do. Traditional accuracy metrics such as MAE, RMSE are considered incomplete because they are unable to evaluate uncertainties. The Continuous Ranked Probability Score (CRPS) generalizes the MAE to evaluate probabilistic forecasts [57]. Given the true observation value x and the cumulative distribution function (CDF) of its forecasts distribution F_X, the CRPS score is given by:

$$\mathrm{CRPS}(F_X, x) = \int_{-\infty}^{\infty} \left(F_X(y) - \mathbb{1}(y-x)\right)^2 \mathrm{d}y \tag{24}$$

where $\mathbb{1}$ is the Heaviside step function. CRPS attains its minimum when the predictive distribution F and the data distribution are identical, such that CRPS is a proper scoring rule [58]. Given n samples $X_i \sim F_X$ as natural samples of the predictive cumulative distribution function (CDF), we employ the empirical CDF of F_X, i.e., $\hat{F_X} = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}\{X_i \leq y\}$.

We also use root-mean-square error (RMSE), which is a standard non-probabilistic evaluation metric. Note the errors are first squared and then averaged, thus the RMSE gives high weight to large errors.

5.2.2 Baselines

5.2.3 Results and analysis

5.3 Forecasting of Events

5.3.1 Data filtering

We use time-series and event sequences extracted from the three datasets as introduced in Section 5.1.1. We first sub-sample these sequences to include at least two events during observing and predicting window $(0, T+\tau]$. This is not necessary for our proposed model as it can deal with situations where no event happens. Most baseline models can only handle event sequences having at least two events. The filtering operation is applied to make different methods comparable. After filtering, there are 1,202 sequences remaining in Electricity dataset, 1087 sequences in Traffic and 3356 sequences in Environment. Each sequence consists 168 hours' past data (time-series and events) and 24 hours' future data for prediction. These sub-sampled datasets are used for probabilistic prediction.

To evaluate deterministic event prediction capability, we need to ensure that there exist events in the predicting window. We then sub-sampled each of the above datasets to include at least 1, 2 and 3 events in their predicting window. Size of datasets after filtering is shown in Table 2. These sub-sampled datasets are used for deterministic multi-step prediction.

Dataset \ # events	1	2	3
Electricity	703	559	371
Traffic	512	469	206
Environment	1627	601	138

TABLE 2: Size (number of sequences) of datasets after filtering to include at least 1, 2 and 3 events in the predicting window.

5.3.2 Baselines

The evaluation involves four peer methods, including two traditional temporal point processes methods and two recurrent neural network(RNN)-based event prediction methods:

Hawkes Process [18]: A Hawkes process captures mutual excitation phenomenon among events. We fit a self-excitation Hawkes process with the conditional intensity function being defined as:

$$\lambda_u(t) = \mu_u + \sum_{i:t_i<t} \alpha_{uu_i}\gamma(t, t_i), \tag{25}$$

where $\gamma(t, t_i) \geq 0$ is the triggering kernel to capture temporal dependencies, μ_u is a background intensity independent of the history for the u-th Hawkes process. We use exponential kernels:

$$\gamma(t, t_i) = \beta_{uu_i}\exp\left(-\beta_{uu_i}(t-t_i)\right) \tag{26}$$

here we fix $\beta_{uu_i} = 0.1$, making it a convex problem so that can be easily optimized [59].

Hawkes Process (regularized) [60]:A multially-exciting multi-dimensional Hawkes process with sparse and low-rank regularization. The objective function of Hawkess-*regularized* is defined as:

$$x \min_{\mathbf{A}\geq 0, \mathbf{u}\geq 0} -\mathcal{L}(\mathbf{A}, \boldsymbol{\mu}) + \lambda_1 \left\|\mathbf{A}\right\|_* + \lambda_2 \left\|\mathbf{A}\right\|_1, \tag{27}$$

where $\mathbf{A} = (\alpha_{uu'})$ for mutually-exciting coefficients and $\boldsymbol{\mu} = (\mu_u)$ for background intensities. The $\mathcal{L}(\mathbf{A}, \boldsymbol{\mu})$ is log-likelihood of Hawkess process model, $\lambda_1 \left\|\mathbf{A}\right\|_*$ is the nuclear norm of matrix $\mathbf{A}$ used to estimate low-rank metrics, and $\lambda_2 \left\|\mathbf{A}\right\|_1$ is the ℓ_1 norm of matrix $\mathbf{A}$ used to enforce sparsity. The parameter λ_1 and λ_2 control the strength of the two regularization terms. We use the same kernel as Hawkess process and set $\lambda_1 = \lambda_2 = 5 \times 10^{-4}$, which is the default value in the standard library.

Hawkes and Hawkes (regularized) processes are implemented by python standard library tick[5].

Recurrent Marked Temproal Point Process (RMTPP) [9]: RMTPP learns the intensity of the marked temporal point process via an RNN; the intensity function is assumed to follow a partially parametric form that can capture non-linear temporal dependencies of future event timing and types on history events. We set the layer and dimension of this RNN the same as Event RNN used in our model: one layer LSTM with 100 neurons.

Attentional Event Recurrent Point Processes (AERPP) [61]: AERPP introduces attention mechanism to RMTPP. Different from RMTPP where hidden states

5. https://x-datainitiative.github.io/tick/index.html

are computed directly by an RNN, AERPP uses attention weights to model the influence strength of past events. We set the structure of AERPP the same as RMTPP and attention function is implemented by Tanh function following [61].

RMTPP and AERPP are from open source as also released by the authors' group[6].

To test how our model leverages correlations between heterogeneous sequences, we introduce three variants of ablated RNN-ODE below.

RNN-ODE (TS ablated): It ablates time-series information of RNN-ODE, making RNN-ODE a generative model which takes past event sequence as input and predicts future events: $p(\mathcal{H}_{T^+:T+\tau}|\mathcal{H}_{0^+:T}, \Phi)$.

RNN-ODE (TS-input ablated): It ablates time-series input of RNN-ODE, making RNN-ODE a generative model which only takes past event sequence as input but predicts future time-series and events: $p(\mathcal{X}_{T+1:T+\tau}, \mathcal{H}_{T^+:T+\tau}|\mathcal{H}_{0^+:T}, \Phi)$.

RNN-ODE (TS-output ablated): It ablates time-series output of RNN-ODE, making RNN-ODE a generative model which takes past time-series and event sequence as input but only predicts future events: $p(\mathcal{H}_{T^+:T+\tau}|\mathcal{X}_{1:T}, \mathcal{H}_{0^+:T}, \Phi)$.

5.3.3 *Evaluation metrics*

We compare the probabilistic and deterministic predicting ability of our model with baselines. For probabilistic predicting, we use Log Likelihood (LL). For deterministic predicting, we use Mean Absolute Error (MAE) for event time prediction and classification Accuracy (ACC) for event type prediction.

Log Likelihood (LL) is a common evaluation metric for temporal point process. It estimates how well the conditional intensity function models a event sequence. Moreover, it does not consider whether there exists event in the predicting window or how many events there are.

Given $\mathcal{H}_{T^+:T+\tau} = \{(c_{n+1}, t_{n+1}), \dots, (c_{n+m}, t_{n+m})\}$, for Hawkes process, Hawkes-*regularized* and RNN-ODE which compute an individual intensity function for each type, LL is computed as:

$$LL(\mathcal{H}_{T^+:T+\tau}) = \sum_{c=1}^{C}(\sum_{j=1}^{m_c} \log \lambda^c(t_j^c) - \int_T^{T+\tau} \lambda^c(t)\mathrm{d}t) \quad (28)$$

For RMTPP which computes overall intensity function and type probability independently, LL is computed as:

$$\begin{aligned} &LL(\mathcal{H}_{T^+:T+\tau}) \\ =&\sum_{j=1}^{m} \log \lambda(t_{n+j}) - \int_T^{T+\tau} \lambda(t)\mathrm{d}t + \sum_{j=1}^{m} \log P(c_{n+j}) \end{aligned} \quad (29)$$

In fact, if we define $\lambda^c(t) = \lambda(t)P(c_{n+j+1} = c), \forall t \in (t_{n+j}, t_{n+j+1}]$, Eq. 29 is consistent with Eq. 28. Yet it is hard to compute Log Likelihood for AERPP, as AERPP is specially designed for deterministic prediction.

For deterministic prediction, we let all methods predict next $s(s = 1, 2, 3)$ event(s) in the predicting window. For our model, this is done by deterministic event prediction algorithm we proposed in Section 4.4.4. For baseline methods, we predict next s events iteratively: predict next event, then use the predicted event as input to predict the next. The timing and the type for these s event(s) are evaluated by MAE and ACC.

6. https://github.com/Thinklab-SJTU/DP3

	Electricity	Traffic	Environment
Hawkes	-4.94	-3.42	-2.01
Hawkes (regularized)	-4.91	-3.34	-2.02
RMTPP	-5.57	-6.49	-4.12
AERPP	/	/	/
RNN-ODE (TS ablated)	-4.85	-2.31	**-1.82**
RNN-ODE (TS-input ablated)	-4.72	-2.28	-1.83
RNN-ODE (TS-output ablated)	-4.47	-1.19	-2.29
RNN-ODE (ours)	**-4.29**	**-1.18**	-2.32

TABLE 3: Evaluation of probabilistic prediction in terms of Log Likelihood (LL) on three datasets.

Mean Absolute Error (MAE) of next s event(s) is computed as:

$$\mathrm{MAE}(\tilde{\mathcal{H}}_{T^+:T+\tau}, \mathcal{H}_{T^+:T+\tau}) = \frac{\sum_{j=1}^{s} |\tilde{t}_{n+j} - t_{n+j}|}{s} \quad (30)$$

Accuracy (ACC) of event type is computed as:

$$\mathrm{ACC}(\tilde{\mathcal{H}}_{T^+:T+\tau}, \mathcal{H}_{T^+:T+\tau}) = \frac{\sum_{j=1}^{s} \mathbb{1}[\tilde{c}_{n+j} = c_{n+j}]}{s} \quad (31)$$

In the above Eq. 30 and Eq. 31, $\tilde{\mathcal{H}}_{T^+:T+\tau} = \{(\tilde{c}_{n+1}, \tilde{t}_{n+1}), \dots, (\tilde{c}_{n+s}, \tilde{t}_{n+s})\}$ denotes the predicted events and $\mathcal{H}_{T^+:T+\tau}$ denotes the ground truth. Note that there may be more than s events in $\mathcal{H}_{T^+:T+\tau}$ but we only care about the first s in the predicting window. Timing is transformed back to the original scale, so MAE falls in $[0, 24]$.

Given a set of sequences, we use mean LL, MAE and ACC for evaluation.

5.3.4 *Results and analysis*

The results of probabilistic and deterministic prediction are shown in Tables 3 and Table 4, respectively. Based on these results, we tend to make the following observations.

Overall performance. The proposed ODE-RNN models beat all baseline models on probabilistic prediction task. For deterministic prediction, our models outperform baseline models significantly on event timing prediction. RMTPP and AERPP perform better than our model on event type prediction, especially for single-step prediction. Compared with baseline methods, the performance of our RNN-ODE models drops more slowly with the increasing prediction steps. In our analysis, such a reduced error accumulation in multi-step prediction is due to the adoption of Neural ODEs instead of iteratively prediction.

Performance of traditional methods. Traditional temporal point process models such as Hawkes and Hawkes (regularized) are still strong benchmarks. As we can see in Table 3, these traditional methods beat neural network-based model RMTPP on probabilistic prediction. Hawkes (regularized) even outperforms RMTPP and AERPP on deterministic timing prediction on Environment dataset. However, due to their strong assumptions about the form of intensity function, it is hard for them to model complex

	Electricity			Traffic			Environment		
Predoction steps	1	2	3	1	2	3	1	2	3
Hawkes	8.30/68.09	10.59/70.98	9.60/50.22	/	/	/	12.69/67.08	11.12/53.33	11.40/47.44
Hawkes (regularized)	9.19/68.79	10.54/**72.77**	10.03/52.89	/	/	/	7.11/53.11	5.15/45.42	9.32/35.90
RMTPP	8.35/**73.44**	7.19/66.41	7.51/**64.58**	7.39/90.63	7.69/73.44	12.64/55.56	10.38/**82.81**	8.29/**75.00**	11.29/34.52
AERPP	8.76/67.19	6.49/64.06	7.20/49.48	5.09/**95.31**	5.05/78.91	11.28/61.90	8.53/79.69	8.76/74.22	10.35/44.05
RNN-ODE (TS ablated)	5.68/57.16	4.95/54.15	**4.16**/49.38	6.39/92.23	6.80/87.5	6.45/57.70	7.01/71.10	5.46/53.18	4.76/41.90
RNN-ODE (TS-input ablated)	5.83/63.76	4.70/62.05	4.35/53.87	6.39/92.23	6.74/**90.21**	6.32/54.76	**5.49**/76.47	5.86/53.88	4.58/50.05
RNN-ODE (TS-output ablated)	**4.84**/65.74	4.29/67.86	4.24/57.24	0.47/92.23	0.85/86.54	**2.62**/80.87	5.84/81.04	5.00/63.26	4.85/49.52
RNN-ODE	4.93/66.31	**4.15**/68.30	4.22/56.67	**0.45**/92.23	**0.81**/84.47	2.76/**81.51**	5.91/79.29	**4.93**/60.74	**4.43**/**50.24**

TABLE 4: Evaluation of deterministic multi-step prediction in terms of timing Mean Absolute Error and type Accuracy(MAE/ACC) on three datasets. Note that Accuracy (ACC) is in percentage.

dynamics of event sequences. Moreover, Hawkes (regularized) performs slightly better than original Hawkes, but not too much. This may be because that Hawkes (regularized) is suitable for high-dimensional ($C > 100$) scene such as social network [60], but dimension of events in our datasets is just 2 or 3.

Performance of RNN-based methods. RMTPP performs even worse than traditional methods on probabilistic prediction. One possible reason is that RMTPP aims to predict timing and type of next event instead of to model the intensity in a whole period of time. Unlike Hawkes process which assumes exponential decay while no event occurs, the intensity of RMTPP may increase exponentially. This can significantly reduce performance when events are sparse, especially when there is no event in the predicting window. Compared with RNN-ODE, RMTPP can be viewed as a semi-parametric model: although it uses RNN to capture the effect of event occurrences, it still makes strong parametric assumptions about the form of intensity between events.

On the other hand, RMTPP assumes event timing and type are (conditionally) independent given history data. Therefore, it can perform well on the task of deterministic type prediction but badly on timing prediction at the same time. This assumption is unrealistic as timing and type have strong correlations in real world and may confuse decision-makers who use the model. By applying attention mechanism to RMTPP, AERPP constantly improves timing prediction accuracy. Both RMTPP and AERPP suffer the problem of error accumulation caused by iteratively prediction: performance decreases quickly with increasing prediction steps. In contrast, RNN-ODE models suffer less from accumulated error by modeling several events in the window as a whole.

Impact of time-series. To investigate the impact of time-series, we also introduce three ablated models. The only difference between RNN-ODE and RNN-ODE (TS-input ablated) is that RNN-ODE takes additional time-series as input(so does RNN-ODE (TS-output ablated) to RNN-ODE (TS ablated)). Tables 3 and 4 show that taking time-series as additional input constantly improves performance. This is intuitive because time-series provides more information even when events are extracted from time-series (Electricity and Traffic).

Finally, we find that predicting time-series together with event sequence slightly improves model performance compared with predicting events alone (RNN-ODE vs TS-output ablated / TS-input ablated vs TS ablated), which can be viewed as a multi-task learning problem. Note that [62] shows that performance of sequence to sequence model can be improved by multi-task learning. This also suggests the correlation between time-series and event sequence.

5.4 Visualization results

future time-series, intensity & ground truth time-series, events w.r.t forecast horizon

6 Conclusion

In this paper, we have formally address the problem of time series forecasting and event sequence prediction given an arbitrary future time window. This setting is of particular practical importance for flexible decision making, while it has been relatively little studied. We present a joint learning framework based on conditional variational autoencoder to formulate the forecasting problem as a conditional sequence generation task, and the Neural ODE is adopted to model the intensity function of temporal point process for event modeling. Extensive experimental results on public benchmarks show the effectiveness of our approach. For future work, we are aimed to improve the scalability of our model on high-dimensional temporal point process, which is a long-standing challenge in temporal point process learning, and it has not been mitigated by neural ODE.

ACKNOWLEDGMENTS

The work is supported by China Major State Research Development Program (2020AAA0107600, 2018YFC0830400), NSFC (U19B2035, 61972250, 72061127003) and SJTU Global Strategic Partnership Fund (2020 SJTU-CORNELL).

REFERENCES

[1] W. W. Wei, "Time series analysis," in *The Oxford Handbook of Quantitative Methods in Psychology: Vol. 2*, 2006.

[2] S. Xiao, J. Yan, X. Yang, H. Zha, and S. M. Chu, "Modeling the intensity function of point process via recurrent neural networks," in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[3] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection for discrete sequences: A survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 24, no. 5, pp. 823–839, 2012.

[4] S. Makridakis, E. Spiliotis, and V. Assimakopoulos, "The m4 competition: Results, findings, conclusion and way forward," *International Journal of Forecasting*, vol. 34, no. 4, pp. 802–808, 2018.

[5] Z. Mariet and V. Kuznetsov, "Foundations of sequence-to-sequence modeling for time series," in *The 22nd International Conference on Artificial Intelligence and Statistics*, 2019, pp. 408–417.

[6] S. Li, X. Jin, Y. Xuan, X. Zhou, W. Chen, Y.-X. Wang, and X. Yan, "Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting," in *Advances in Neural Information Processing Systems*, 2019, pp. 5244–5254.

[7] S. S. Rangapuram, M. W. Seeger, J. Gasthaus, L. Stella, Y. Wang, and T. Januschowski, "Deep state space models for time series forecasting," in *Advances in Neural Information Processing Systems*, 2018, pp. 7785–7794.

[8] D. Salinas, V. Flunkert, J. Gasthaus, and T. Januschowski, "Deepar: Probabilistic forecasting with autoregressive recurrent networks," *International Journal of Forecasting*, 2019.

[9] N. Du, H. Dai, R. Trivedi, U. Upadhyay, M. Gomez-Rodriguez, and L. 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*. ACM, 2016, pp. 1555–1564.

[10] E. Bacry, I. Mastromatteo, and J.-F. Muzy, "Hawkes processes in finance," *Market Microstructure and Liquidity*, vol. 1, no. 01, p. 1550005, 2015.

[11] S. Xiao, J. Yan, C. Li, B. Jin, X. Wang, X. Yang, S. M. Chu, and H. Zha, "On modeling and predicting individual paper citation count over time." in *IJCAI*, 2016, pp. 2676–2682.

[12] W. Wu, J. Yan, X. Yang, and H. Zha, "Decoupled learning for factorial marked temporal point processes," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2018, pp. 2516–2525.

[13] M. Bińkowski, G. Marti, and P. Donnat, "Autoregressive convolutional neural networks for asynchronous time series," *arXiv preprint arXiv:1703.04122*, 2017.

[14] L. Li, J. Zhang, Y. Zhang, J. Yan, Y. Jin, Y. Duan, and G. Tian, "Synergetic learning of heterogeneous temporal sequences for multi-horizon probabilistic forecasting," in *AAAI*, 2021.

[15] O. Aalen, O. Borgan, and H. Gjessing, *Survival and event history analysis: a process point of view*. Springer Science & Business Media, 2008.

[16] D. J. Daley and D. Vere-Jones, *An introduction to the theory of point processes: volume II: general theory and structure*. Springer Science & Business Media, 2007.

[17] Y. Ogata, "Space-time point-process models for earthquake occurrences," *Annals of the Institute of Statistical Mathematics*, vol. 50, no. 2, pp. 379–402, 1998.

[18] A. G. Hawkes, "Spectra of some self-exciting and mutually exciting point processes," *Biometrika*, vol. 58, no. 1, pp. 83–90, 1971.

[19] M. Gomez-Rodriguez, D. Balduzzi, and B. Schölkopf, "Uncovering the temporal dynamics of diffusion networks," in *Proceedings of the 28th International Conference on International Conference on Machine Learning*, 2011, pp. 561–568.

[20] Q. Zhao, M. A. Erdogdu, H. Y. He, A. Rajaraman, and J. Leskovec, "Seismic: A self-exciting point process model for predicting tweet popularity," in *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, 2015, pp. 1513–1522.

[21] H. Mei and J. M. Eisner, "The neural hawkes process: A neurally self-modulating multivariate point process," in *Advances in Neural Information Processing Systems*, 2017, pp. 6754–6764.

[22] R. T. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, "Neural ordinary differential equations," in *Advances in neural information processing systems*, 2018, pp. 6571–6583.

[23] Y. Rubanova, R. T. Chen, and D. Duvenaud, "Latent odes for irregularly-sampled time series," *arXiv preprint arXiv:1907.03907*, 2019.

[24] J. Jia and A. R. Benson, "Neural jump stochastic differential equations," in *Advances in Neural Information Processing Systems*, 2019, pp. 9847–9858.

[25] H. van Houwelingen and H. Putter, *Dynamic prediction in clinical survival analysis*. CRC Press, 2011.

[26] D. R. Cox, "Regression models and life-tables," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 34, no. 2, pp. 187–202, 1972.

[27] J. D. Kalbfleisch and R. L. Prentice, *The statistical analysis of failure time data*. John Wiley & Sons, 2011, vol. 360.

[28] H. Soleimani, J. Hensman, and S. Saria, "Scalable joint models for reliable uncertainty-aware event prediction," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 8, pp. 1948–1963, 2017.

[29] J. F. Tierney, L. A. Stewart, D. Ghersi, S. Burdett, and M. R. Sydes, "Practical methods for incorporating summary time-to-event data into meta-analysis," *Trials*, vol. 8, no. 1, p. 16, 2007.

[30] P. R. Williamson, C. T. Smith, J. L. Hutton, and A. G. Marson, "Aggregate data meta-analysis with time-to-event outcomes," *Statistics in medicine*, vol. 21, no. 22, pp. 3337–3351, 2002.

[31] P. Wang, T. Shi, and C. K. Reddy, "Tensor-based temporal multi-task survival analysis," *IEEE Transactions on Knowledge and Data Engineering*, 2020.

[32] J. L. Katzman, U. Shaham, A. Cloninger, J. Bates, T. Jiang, and Y. Kluger, "Deepsurv: personalized treatment recommender system using a cox proportional hazards deep neural network," *BMC medical research methodology*, vol. 18, no. 1, p. 24, 2018.

[33] G. A. Bello, T. J. Dawes, J. Duan, C. Biffi, A. de Marvao, L. S. Howard, J. S. R. Gibbs, M. R. Wilkins, S. A. Cook, D. Rueckert *et al.*, "Deep-learning cardiac motion analysis for human survival prediction," *Nature machine intelligence*, vol. 1, no. 2, pp. 95–104, 2019.

[34] H. Kvamme, Ø. Borgan, and I. Scheel, "Time-to-event prediction with neural networks and cox regression," *Journal of machine learning research*, vol. 20, no. 129, pp. 1–30, 2019.

[35] C. Lee, W. R. Zame, J. Yoon, and M. van der Schaar, "Deephit: A deep learning approach to survival analysis with competing risks." in *AAAI*, 2018, pp. 2314–2321.

[36] M. Sudell, R. Kolamunnage-Dona, and C. Tudur-Smith, "Joint models for longitudinal and time-to-event data: a review of reporting quality with a view to meta-analysis," *BMC medical research methodology*, vol. 16, no. 1, p. 168, 2016.

[37] A. A. Tsiatis and M. Davidian, "Joint modeling of longitudinal and time-to-event data: an overview," *Statistica Sinica*, pp. 809–834, 2004.

[38] D. Collett, *Modelling survival data in medical research*. CRC press, 2015.

[39] G. E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[40] R. Hyndman, A. B. Koehler, J. K. Ord, and R. D. Snyder, *Forecasting with exponential smoothing: the state space approach*. Springer Science & Business Media, 2008.

[41] H. Lütkepohl, *New introduction to multiple time series analysis*. Springer Science & Business Media, 2005.

[42] D. Salinas, M. Bohlke-Schneider, L. Callot, R. Medico, and J. Gasthaus, "High-dimensional multivariate forecasting with low-rank gaussian copula processes," in *Advances in Neural Information Processing Systems*, 2019, pp. 6824–6834.

[43] L. Li, J. Yan, X. Yang, and Y. Jin, "Learning interpretable deep state space model for probabilistic time series forecasting," in *Proceedings of the 28th International Joint Conference on Artificial Intelligence*. AAAI Press, 2019, pp. 2901–2908.

[44] Y. Wang, A. Smola, D. C. Maddix, J. Gasthaus, D. Foster, and T. Januschowski, "Deep factors for forecasting," *arXiv preprint arXiv:1905.12417*, 2019.

[45] I. Sutskever, O. Vinyals, and Q. Le, "Sequence to sequence learning with neural networks," *Advances in NIPS*, 2014.

[46] C. Fan, Y. Zhang, Y. Pan, X. Li, C. Zhang, R. Yuan, D. Wu, W. Wang, J. Pei, and H. Huang, "Multi-horizon time series forecasting with temporal attention learning," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2019, pp. 2527–2535.

[47] L. Vincent and N. Thome, "Shape and time distortion loss for training deep time series forecasting models," in *Advances in Neural Information Processing Systems*, 2019, pp. 4191–4203.

[48] S. B. Taieb and A. F. Atiya, "A bias and variance analysis for multistep-ahead time series forecasting," *IEEE transactions on neural networks and learning systems*, vol. 27, no. 1, pp. 62–76, 2015.

[49] Z. Che, S. Purushotham, G. Li, B. Jiang, and Y. Liu, "Hierarchical deep generative models for multi-rate multivariate time series," in *International Conference on Machine Learning*, 2018, pp. 783–792.

[50] I. M. Baytas, C. Xiao, X. Zhang, F. Wang, A. K. Jain, and J. Zhou, "Patient subtyping via time-aware lstm networks," in *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*. ACM, 2017, pp. 65–74.

[51] E. De Brouwer, J. Simm, A. Arany, and Y. Moreau, "Gru-ode-bayes: Continuous modeling of sporadically-observed time series," in *Advances in Neural Information Processing Systems*, 2019, pp. 7379–7390.

[52] K. Sohn, H. Lee, and X. Yan, "Learning structured output representation using deep conditional generative models," in *Advances in neural information processing systems*, 2015, pp. 3483–3491.

[53] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[54] A. Pagnoni, K. Liu, and S. Li, "Conditional variational autoencoder for neural machine translation," *arXiv preprint arXiv:1812.04405*, 2018.

[55] T. Wang and X. Wan, "T-cvae: Transformer-based conditioned variational autoencoder for story completion." in *IJCAI*, 2019, pp. 5233–5239.

[56] D. Kingma and M. Welling, "Auto-encoding variational bayes," in *arXiv:1312.6114*, 2013.

[57] T. Gneiting and M. Katzfuss, "Probabilistic forecasting," *Annual Review of Statistics and Its Application*, vol. 1, pp. 125–151, 2014.

[58] T. Gneiting and A. E. Raftery, "Strictly proper scoring rules, prediction, and estimation," *Journal of the American Statistical Association*, vol. 102, no. 477, pp. 359–378, 2007.

[59] E. Bacry, I. Mastromatteo, and J.-F. Muzy, "Hawkes processes in finance," *Market Microstructure and Liquidity*, vol. 01, no. 01, p. 1550005, 2015.

[60] K. Zhou, H. Zha, and L. Song, "Learning social infectivity in sparse low-rank networks using multi-dimensional hawkes processes," in *Artificial Intelligence and Statistics*, 2013, pp. 641–649.

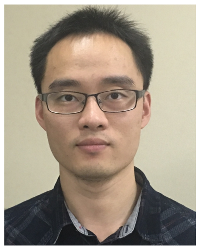
[61] S. Xiao, J. Yan, M. Farajtabar, L. Song, X. Yang, and H. Zha, "Learning time series associated event sequences with recurrent point process networks," *IEEE transactions on neural networks and learning systems*, 2019.

[62] M.-T. Luong, Q. V. Le, I. Sutskever, O. Vinyals, and L. Kaiser, "Multi-task sequence to sequence learning," 2016.

Longyuan Li received the B.E. degree in Electronic Engineering from Huazhong University of Science and Technology in 2015. He is currently working toward the Ph.D. degree in the Department of Electronic Engineering, Shanghai Jiao Tong University, China. His research interests include machine learning and data mining. He focuses on probabilistic models and Bayesian non-parametric models, for sequential data such as multi-dimensional time-series. He is also interested in deep probabilistic graphical models and uncertainty estimation. He has published first-authored papers in AAAI, IJCAI, and IEEE TNNLS.

Yunhao Zhang is a senior undergraduate in the department of computer science and engineering, Shanghai Jiao Tong University. His research interests include deep learning and data mining. His recent research is mainly focused on sequential data modeling, especially for temporal sequence forecasting and analysis.

Jihai Zhang is a senior undergraduate in the department of computer science and engineering, Shanghai Jiao Tong University, China. His research interests include machine learning and data mining. He focuses on time series processing, including deep clustering, probabilistic forecasting and representation learning. He is also interested in generative models and contrastive learning. He has published first-authored paper in AAAI.



Junchi Yan (M'10) is currently an Associate Professor with Shanghai Jiao Tong University. Before that, he was a Senior Research Staff Member and Principal Scientist for industrial vision with IBM Research where he started his career since April 2011. He obtained the Ph.D. at the Department of Electronic Engineering of Shanghai Jiao Tong University, China. He received the ACM China Doctoral Dissertation Nomination Award and China Computer Federation Doctoral Dissertation Award. His research interests are machine learning and visual computing. He serves as an Associate Editor for IEEE ACCESS, Managing Guest Editor for IEEE Transactions on Neural Networks and Learning Systems, Pattern Recognition Letters, and Area Chair for CVPR21, ICPR20, Senior PC for IJCAI21, CIKM19. He is the awardee of IBM Journal of Eminence and a Member of IEEE.

Yaohui Jin was once a Technical Staff Member with Bell Labs Research China. After that he joined Shanghai Jiao Tong University in 2002, where he is a Professor with the State Key Laboratory of Advanced Optical Communication Systems and Networks and the Deputy Director of Network and Information Center. His research interests include civic engagement and open innovation, cloud computing network architecture, and streaming data analysis. He is the Founder of OMNILab, which is an open innovation lab focusing on data analysis. He has published over 100 technical papers in leading conferences and journals and is the owner of over 10 patents. In 2014, OMNILab won the champion of CCF national big data challenge among nearly 1000 teams, and was the champion of the Shanghai open data innovation and creation competition. He has served over 10 technical committees. He enthuses public service and science popularization, actively promotes crowd engaged innovation, and interdisciplinary collaboration.

Xiaokang Yang (M'00-SM'04-F'19) received the B. S. degree from Xiamen University, in 1994, the M. S. degree from Chinese Academy of Sciences in 1997, and the Ph.D. degree from Shanghai Jiao Tong University in 2000. He is currently a Distinguished Professor of School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai, China. His research interests include visual signal processing and communication, media analysis and retrieval, and pattern recognition. He serves as an Associate Editor of IEEE Transactions on Multimedia and an Associate Editor of IEEE Signal Processing Letters. Prof. Yang is also a fellow of IEEE.