# Bellman Meets Hawkes: Model-Based Reinforcement Learning via Temporal Point Processes

**Chao Qu*[1], Xiaoyu Tan*† [1], Siqiao Xue[1], Xiaoming Shi[1], James Zhang[1], Hongyuan Mei[2]**
[1] Ant Group, Hangzhou, China.
[2] Toyota Technological Institute at Chicago, Chicago, IL, United States.
chaoqu.technion@gmail.com,txywilliam1993@outlook.com, {siqiao.xsq,peter.sxm,james.z}@antgroup.com
hongyuan@ttic.edu

## Abstract

We consider a sequential decision making problem where the agent faces the environment characterized by the stochastic discrete events and seeks an optimal intervention policy such that its long-term reward is maximized. This problem exists ubiquitously in social media, finance and health informatics but is rarely investigated by the conventional research in reinforcement learning. To this end, we present a novel framework of the model-based reinforcement learning where the agent's actions and observations are asynchronous stochastic discrete events occurring in continuous-time. We model the dynamics of the environment by Hawkes process with external intervention control term and develop an algorithm to embed such process in the Bellman equation which guides the direction of the value gradient. We demonstrate the superiority of our method in both synthetic simulator and real-data experiments.

## Introduction

The last several years have witnessed the great success of reinforcement learning (RL) including the video game playing (Mnih et al. 2015), robot manipulation (Gu et al. 2017), autonomous driving (Shalev-Shwartz, Shammah, and Shashua 2016) and many others (Lazic et al. 2018; Dalal, Gilboa, and Mannor 2016). Most of them focus on the problem where the system of interest evolves continuously with time, e.g., a trajectory of a robot arm.

However, the conventional research in RL may omit a category of system that evolves continuously and may be interrupted by stochastic events *abruptly* ( See the jumps at $t_1, t_2, t_3$ in Figure 1). Such system exists ubiquitously in the social and information science and therefore necessitates the research of reinforcement learning in these domains to extend its applicability in the real-world problems (Farajtabar et al. 2017; Wang et al. 2018), in which the agent seeks an optimal intervention policy so as to improve the future course of events. Concrete examples may include:

- Social media. Social media website allows users to create and share content. Retweet can form as users resharing and broadcasting others' tweet to their friends and followers. Such stochastic events would steer the behaviors of other tweet users (Rizoiu et al. 2017); the platform (agent) may want to seek a policy to effectively mitigate the fake news by optimizing the performance of real news propagation over the network (Farajtabar et al. 2017).
- Medical events. The stream of medical events would include the blood test, diagnosis, treatment and so on. To cure the patients, a doctor (the agent) often seeks the optimal treatments based on the clinical conditions.

In the above problems, given the observed trajectories of the event sequences in continuous-time, the agent can impose the intervention on the environment to maximize the long term reward. For instance, the platform can post the valid news in the social media to steer the spontaneous user mitigation activities of fake news. In the RL context, the action is to post valid news and the observation is the users' following behaviors such as retweet while the reward can be quantified by event exposure counts. However, the following characteristics of event sequences bring new challenges:

1. The inter-event time intervals are *not* constant while conventional RL considers the synchronized action and observations. For instance, the healthy patient with less severe symptoms would visit doctors occasionally, while the unhealthy patients receive the test and treatment more frequently. Thus it is crucial to predict both which events are likely to happen next and when they will happen. Directly adapting conventional RL algorithm to its Semi-MDP (SMDP) version can not address this issue well (see our *ablation study* in Appendix. )
2. How to describe the dynamics of an abrupt event if the agent imposes intervention? Note that these events may help cause or prevent future events in a complex mechanism, e.g., in Figure 1, the platform posts the valid news to steer the spontaneous user activities. In general, the valid news and fake news can inhibit each other. Specifically, the dynamic system characterized by continuous trajectories will be abruptly changed by discrete events. However, the dynamic models of most model-based RL methods are continuous *without* endogenous jumps (e.g., the jump corresponding to the spontaneous user activities), and therefore are not suitable for learning the continuous and discrete dynamics of such hybrid system.

*These authors contributed equally.
†Corresponding author. txywilliam1993@outlook.com

To this end, we propose a novel model-based reinforcement learning, where the agent seeks the long term reward by imposing the interventions (event) on this hybrid system. In our setting, both the actions taken by the agent and the observations from the environment are asynchronous stochastic event in continuous time. To describe the dynamics of the environment, we adapt Neural Hawkes Process (Zhang et al. 2020; Zuo et al. 2020; Mei and Eisner 2017) to the case with exogenous Interventions (i.e., the actions in RL) and name it NHPI for short. Instead of discretizing time into fixed-width intervals in conventional RL, we treat timestamps as random variables. The latent representations of NHPI implicitly describe when and which type of event will happen in the future by learning the intensity function $\lambda$. We embed the latent space of the learned dynamic model into the Bellman equation and derive a stochastic value gradient algorithm in the semi-Markov decision process to cope with the challenge of irregular timestamps.

It is known that Hawkes process can be reformulated as stochastic differential equations (SDEs) *with endogenous jumps* caused by stochastic events inside the environment itself. The dynamic model of most traditional model-based RL methods can be described by SDEs or ODEs *without* endogenous jumps (if we adapt them in the continuous-time setting). Thus, it reveals the intrinsic difference between our approach and the previous model-based RL methods, which is evidenced by the experimental results showing that our algorithm outperforms model-free and model-based RL baselines with a notable margin by considering the discontinuity nature of the jumps. We discuss the connections to ODE approach and world model (Hafner et al. 2019a) thoroughly in Appendix.

**Contributions:** We propose a model-based RL algorithm to solve the problem that can evolve both continuously and can be interrupted by the stochastic event abruptly. Our method captures the jump and the irregular time-interval property in the systems and extends the applicability of RL in the area of social media, medical events and many others. Empirically, we demonstrate that our method surpasses the performances of other state-of-the-art model-free and model-based RL algorithms. We release our code at https://github.com/WilliamBUG/Event_driven_rl/tree/main

## Related Work

Temporal point processes are prevalent in modeling the sequences of events happening at irregular intervals and have wide applications in e-commerce systems, social media, and traffic prediction (Hawkes and Oakes 1974; Daley and Vere-Jones 2007). Recent works combine temporal point process with deep learning and design algorithms to learn complex behavior from real-world data (Mei and Eisner 2017; Du et al. 2016; Zuo et al. 2020; Zhang et al. 2020; Xue et al. 2021, 2022). However, these works just focus on the *prediction* problem rather than the *control* problem, i.e., the agent does not make any decision to impose the intervention on the environment. The only exceptions are (Wang et al. 2018; Zarezade et al. 2017; Upadhyay, De, and Gomez-Rodriguez 2018; Farajtabar et al. 2017). The problem is formulated into SDEs in (Wang et al. 2018; Zarezade et al. 2017) and solved with the optimal control theory. However they assume that the dynamic are known, which is not applicable in most real-world problems. TPPRL proposes a modified version of REINFORCE algorithm (Upadhyay, De, and Gomez-Rodriguez 2018). In general, such simple model-free algorithm suffers from high sample-complexity. The authors in (Farajtabar et al. 2017) use linear least square temporal difference (LSTD) learning to learn a policy to mitigate the fake news over the social media. However, it can only apply to simple problem due to the limitation of LSTD. Our work builds a flexible model to describe the underlying dynamics of the event stream intervened with the action, and then we propose a sample-efficient model-based RL algorithm where the agent seeks the optimal policy to achieve some desired states and to maximize the long-term reward.

There are a few works in literature on the Semi-MDPs and the continuous-time reinforcement learning (Munos 2006; Bradtke and Duff 1995), which are natural formalism for the problem with irregular time intervals. However, these works are generally model-free algorithm and have low sample efficiency. The problem in social science is partially observable in the real-world, which made RNN and its variants popular to summarize the long-term dependencies and describe the belief states in RL (Oh et al. 2015). In model-based approaches (Ha and Schmidhuber 2018; Hafner et al. 2019a,b), a world model is often learned from the past experience and generates new imaginary data to feed the agent. These algorithms also have downsides, i.e., they generally discretize the time evenly. The authors in (Du, Futoma, and Doshi-Velez 2020; Yıldız, Heinonen, and Lähdesmäki 2021) use neural ODE to create the world model and hence can cope with the continuous-time setting. However their work can not handle the hybrid dynamic with jumps and therefore their empirical results focus on the Openai gym (e.g., swimmer) (Brockman et al. 2016). As shown in Appendix, our model can be thought of as SDEs with jumps. To the best of our knowledge, none work mentioned above (except ours) considers the dynamics with jumps sparked by the event, which is essential in modeling real-world problems, since such jump term related to the counting measure reflects the nature in the social science, epidemic process and many others (Cox and Isham 1980; Rasmussen 2018). In our experimental result, we also demonstrate our algorithm outperforms the state of the arts.

## Preliminary

As we discussed in the introduction, we need to properly handle the problem with irregular time interval which is caused by the abrupt event. To this end, we introduce temporal point process and the Semi-MDPs, which are building blocks of our work.

**Temporal Point Process:** A temporal point process (TPP) is a random process whose realization consists of a list of discrete events localized in time $\mathcal{T} = \{t_1, ..., t_N\}$, where $t_i < t_{i+1}$ (Daley and Vere-Jones 2007). The marked temporal point process is a stochastic process whose realization consists not only the event time but also event type $k \in \mathcal{K}$, where $\mathcal{K} := \{1, ..., K\}$. Let $Es = \{(t_i, k_i)\}_{i=1}^{L}$ be an event

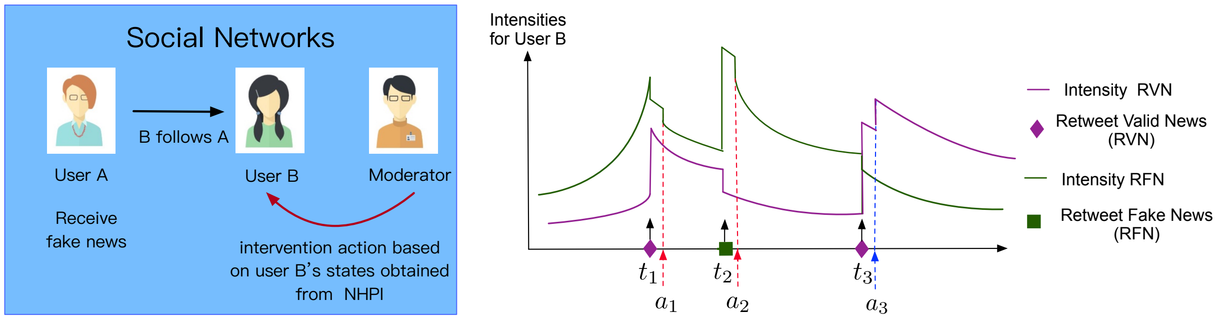


Figure 1: The figure in the left panel illustrates the fake new mitigation problem. The user B may follow the fake news from user A and spread it to others. The moderator may expose valid news to user B to incentivize more spontaneous mitigation events (e.g., valid news at $t_3$). The right panel is the corresponding intensity function of valid news and the fake news modeled by the Neural Hawkes Process with intervention. The probability that an event occurs in the infinitesimal interval $[t, t+dt]$ is determined by intensity times $dt$ (i.e., $\lambda_i(t)dt$) as that in TPP. Events of valid news and fake news inhibit each other. When events occur at time $t_1, t_2, t_3$, there are immediate effects (the sudden jumps) and long-term effects (decay with time) in intensity. In addition, the agent has two types of actions to intervene in the event stream: red action to inhibit fake news event and blue one to excite valid news event, which cause instant effects on the intensity of events. In our setting, the agent takes actions at timestamp $t_i$ or with a small delay (see the section of problem setup ).

sequence with length $L$, where $k_i$ is the $i$-th event of the sequence $Es$ and $t_i$ is the timestamp of the $i$-th event. Then the history is defined as $\mathcal{H}_t := \{(t_i, k_i)|t_i < t, k_i \in \mathcal{K}\}$. In practice, $k_i$ can represent the messages posted in the social media, or the treatment given by the doctor while $t_i$ is the corresponding occurring time of such event. We use $\lambda_k(t)$ to represent the rate of an event with type $k$ and denote $\lambda(t) = \sum_{k=1}^{K} \lambda_k(t)$ as the total intensity of all events. Hawkes process captures the *interactions* between different types of event, in which past events conspire to raise the intensity of each type of event (Hawkes and Oakes 1974). In particular, the intensity with type $k$ is modeled as:$\lambda_k(t) = \mu_k + \sum_{h:t_h<t} \beta_{k_h,k} \exp(-\zeta(t-t_h))$ where $\mu_k$ is the base intensity of event type $k$, $\beta_{j,k} \geq 0$ is the degree to which an event of type $j$ excites type $k$, and $\zeta > 0$ is the decay rate of that excitation. Random sequence can be generated from the model by the thinning algorithm (Lewis and Shedler 1979). Recent works (Mei and Eisner 2017; Zuo et al. 2020; Zhang et al. 2020) combine the progress in deep learning to enhance the capacity of this process, e.g., using RNN or transformer to approximate the intensity function and can predict online and off-line human actions accurately in social media.

**Semi-MDPs:** A semi-Markov decision process (SMDP) is a tuple $(\mathcal{S}, \mathcal{A}, P, \mathcal{R}, \mathcal{T}, F, \rho)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{T}$ is a transition time space, $\mathcal{P}$ is the transition probability, and $e^{-\rho}$ is the discount factor with $\rho > 0$ (Bradtke and Duff 1995). A Semi-Markov process evolves as follows. The next state $s'$ is chosen according to the transition probabilities $P(s'|s,a)$. Conditioned on the event that the next state is $s'$, the time interval $\tau \in \mathcal{T}$ between the $s$ to $s'$ has the probability distribution of $F(\cdot|a,s,s')$. The reward function $r(t) := R(s(t), a(t))$. The objective of the agent is to find a policy $\pi(a|s)$ to maximize the expected infinite horizon discounted reward $\mathbb{E}[\int_0^\infty e^{-\rho t} r(t)dt]$. The Bellman equation for the SMDPs is defined as follows (Bradtke and Duff 1995).

$$V(s) = \sum_{a,s'} \pi(a|s) P(s'|a,s) [\int_0^\infty \int_0^t e^{-\rho\tau} r(\tau) d\tau dF(t|s,a,s') + \int_0^\infty e^{-\rho t} V(s') dF(t|s,a,s')]. \tag{1}$$

In contrast to the constant time interval in MDP, $\tau$ in SMDP is stochastic (Sutton, Precup, and Singh 1999). Our high level idea is to model such interval with the temporal point processes so as to describe the dynamics of the control problem in SMDPs. We provide some concrete examples in the social media problem to relate these definitions to the real world problems in the following. The state $s$ is the count of the events with different types, e.g., the numbers of fakes and real news received by the users, or the embedding of these events sequence in the *hidden space* of RNN. The action $a$ is the intervention, e.g., the network moderator exposes different types of the real news (actions) to users to match their exposure to fake news. The reward can be the distance between the target frequency of the event and the current frequency, and the cost of different actions can be different.

## Our Method

This section is organized as follows: we first describes the setting of the problem. Then we modify the Hawkes process to incorporate the effect of actions. Using NHPI, we derive a model-based stochastic value gradient algorithm for RL in the SMDPs setting. We relate this model to the SDEs with jumps and discuss the key difference between our reinforcement learning algorithm and traditional RL through our SDE reformulation in Appendix.

### Problem Setup

In many problems, including social media, medical events and many others, the agent observes a steam of the asynchronous stochastic discrete events occurring in continuous-time, such as the user's posts, clicks or likes. We denote the observations from the environment $\mathcal{H}_{\leq t_i}$ up to $t_i$ (including $t_i$) as $\{(t_1, k_1), ..., (t_i, k_i)\}$. In our setting, the agent can just intervene in the environment or change its actions $a_i$ at timestamp $t_i$. We denote its action sequence before time $t_i$ (not including $t_i$) as $a_{<t_i}$. At timestamp $t_i$, the agent samples its action from a policy, i.e., $a_i \sim \pi(\cdot|\mathcal{H}_{\leq t_i}, a_{<t_i})$. Unless otherwise specified, we assume $\mathcal{A} \subseteq \mathcal{K}$, i.e., the action set is a subset of the total event set. After imposing the intervention on the environment, the agent obtains a reward $r(t)$ and aims to maximize the discounted long-term reward, $\mathbb{E}[\int_0^\infty e^{-\rho t} r(t) dt]$, where the expectation is over the randomness of the transition, time interval and reward in the SMDPs. In practice, the agent may apply the action $a_i$ after $t_i$ with a small delay $\delta t$ (see the illustration in Figure 1). For instance, after realizing that rumors spreads over the social media, the network moderator may need some response time $\delta t$ to apply the strategy to block the diffusion. The doctor provide the treatment for the patient after the medical test. This $\delta t$ can be arbitrary predefined positive value in accordance with tasks. In our experiment, we impose the intervention at the next coming clock tick for simplicity. To ease our notion here, we simply assume $a_i$ occurs at time $t_i$.

## NHPI Model

Recall that our high level idea is to leverage the Hawkes process to model *time intervals* $\tau$ in SMDPs so as to describe the underlying dynamics of stochastic event. In the following, we demonstrate how to modify the neural Hawkes process to the RL setting. In particular, we build a dynamic model following the recent progress in attention (Vaswani et al. 2017) and neural Hawkes process (Mei and Eisner 2017; Zhang et al. 2020; Zuo et al. 2020). It is natural to extend the definition of the event stream in the section of preliminary to the event-action stream $Es = \{(t_i, k_i, a_i)\}$ to incorporate exogenous intervention $a$. The whole structure is summarized in Figure 2, where the latent variable $\boldsymbol{h}(t_i)$ is a key vector which summarizes the history and decides the intensity $\lambda(t_i)$ of the Hawkes process. Then the latent state feeds into the agent as the state information. Besides the policy function and value function, the agent builds the transition and reward model over the latent state .

*Encode observations to hidden state* $\mathbf{h}$: As the common practice in RL (Ha and Schmidhuber 2018) and TPP (Mei and Eisner 2017), we encode the observations, i.e. the sequence of event types, timestamps and actions $\{(t_i, k_i, a_i)\}_{i=1}^L$ into dense embedding $\mathbf{X} \in \mathbb{R}^{L \times N}$ in figure 2. Then such embedding passes through Multi-head attention, $\mathbf{\Lambda} = softmax(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{M_K}})\mathbf{V}$ where $\mathbf{Q} = \mathbf{X}\mathbf{W}^Q$, $\mathbf{K} = \mathbf{X}\mathbf{W}^K, \mathbf{V} = \mathbf{X}\mathbf{W}^V$ are the query, key and value matrix respectively as that in (Vaswani et al. 2017; Zuo et al. 2020). $\mathbf{W}^Q, \mathbf{W}^K \in \mathbb{R}^{N \times M_K}$, $\mathbf{W}^V \in \mathbb{R}^{N \times M_V}$ are learned weight matrix. To avoid observing the future, the attention is equipped with masks. When computing the attention $\mathbf{S}(j, :)$, we mask all the future position. Then we feed the self-attention $\mathbf{\Lambda}$ into a feed-forward neural network to generate the hidden representation $\mathbf{H}$. The hidden representation at

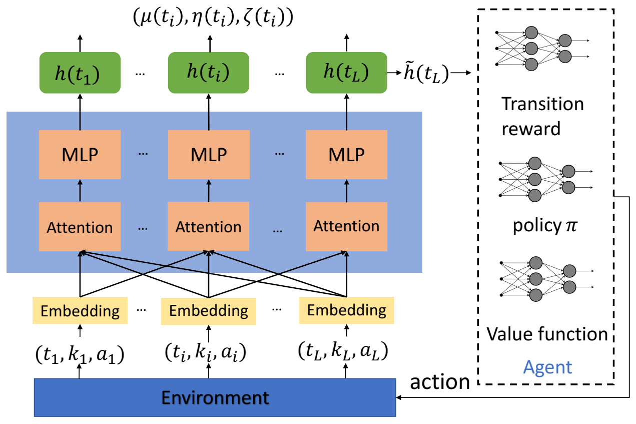


Figure 2: Structure of NHPI. We embed the $(t_i, k_i, a_i)$ into a dense vector, and pass two multi-head attention+ feed forward modules with masks. The outputs are $\mathbf{h}(t_i), \boldsymbol{\mu}(t_i), \boldsymbol{\eta}(t_i), \boldsymbol{\zeta}(t_i), \boldsymbol{\lambda}(t_i)$. We then feed states, i.e., the latent representation $\tilde{\boldsymbol{h}}$ to our RL module.

time step $t_i$ is the $i$-th column of $\mathbf{H}$, i.e., $\mathbf{h}(t_i) = \mathbf{H}(i, :)$. We defer the description of how to construct embedding $\mathbf{X}$ and parameterizations of neural networks to Appendix.

*Construct intensity function* $\lambda_k(t)$*:* We follow the work in (Zhang et al. 2020) to define the intensity function. $\lambda_k(t) = softplus(\mu_k(t_i) + \big(\eta_k(t_i) - \mu_k(t_i)\big) \times \exp\big(-\zeta_k(t_i) \times (t - t_i)\big)$, where $\mu_k(t_i) = Relu(\mathbf{w}_{\mu,k}^T \mathbf{h}(t_i)), \eta_k(t_i) = Relu(\mathbf{w}_{\eta,k}^T \mathbf{h}(t_i))$, $\zeta_k(t_i) = softplus(\mathbf{w}_{\zeta,k}^T \mathbf{h}(t_i))$ and $\mathbf{w}_{\mu,k}^T, \mathbf{w}_{\eta,k}^T, \mathbf{w}_{\zeta,k}^T$ are learned vectors. Notice that now the intensity also depends on the historical action sequence.

## Training NHPI Model

Given NHPI, the next step is to define the training objective. In the following, we provide the likelihood of the model defined as the density function of $(t_i, k_i)$ given the intervention $a_i$ in the event stream $\{(t_i, k_i, a_i)\}_{i=1}^L$ with $t_L < T$. The Hawkes process has a compact form of the log-likelihood function which only depends on $\lambda$. The log-likelihood of Hawkes process given observations in $[0, T)$ is $\sum_{i:t_i<T} \log \lambda_{k_i}(t_i) - \int_{t=0}^T \lambda(t) dt$, where $\lambda(t)$ is the sum intensity of all event types, i.e., $\lambda(t) = \sum_{k=1}^K \lambda_k(t)$ (Rasmussen 2018). We show NHPI has the same form of the log-likelihood function by adapting the proof in Hawkes process to allow external action and include the proof in Appendix for the completeness. The term $\sum_{i:t_i<T} \log \lambda_{k_i}(t_i)$ is easy to obtain from samples, since $\lambda_k(t_i), k \in \{1, ..., K\}$ are the outputs from our model. Specifically, we need to conduct a Monte-Carlo sampling to estimate the integral term and we will defer the details to appendix. Thus the whole objective function can be optimized by stochastic gradient descent.

## Planning over the Latent Space of NHPI

**State**: Recall that $\mathbf{h}(t_i)$ encodes the history of events and actions $\{(t_j, k_j, a_j)\}_{j=1,..,i}$, e.g., the history of fakes/real news retweeted by the user and mitigation valid news posted by the platform, which is generally be used as belief state in

RL (Hafner et al. 2019a). We define another vector $\tilde{\mathbf{h}}(t_i)$, which encodes the same history but toggle the last action $a_i = \mathbf{0}$. It is an embedding of history before an action at timestamp $t_i$ (i.e., $\mathcal{H}_{\leq t_i}, a_{<t_i}$). In our current algorithm, we directly treat $\tilde{\mathbf{h}}(t_i)$ as our state $s(t_i)$, i.e., $s(t_i) := \tilde{\mathbf{h}}(t_i)$ but other transformations on $\tilde{\mathbf{h}}$ can also be accommodated.

**Transition learning**: Given NHPI, we can obtain state $s$ (i.e., $\tilde{\mathbf{h}}$) at every timestamp. Then we use a neural network $g$ to learn the transition such that $s(t_{i+1}) \approx g(s(t_i), a_i, \epsilon(s(t_i))$, where $\epsilon$ is the noise term to represent the stochasticity in the transition $P(s(t_{i+1})|s(t_i), a_i)$ of SMDP . In particular, we minimize the error

$$\|s(t_{i+1}) - g(s(t_i), a_i, \epsilon(s_{t_i}))\|_2^2, \tag{2}$$

where $g$ is a neural network with reparameterization trick (Kingma and Welling 2013) to incorporate the noise term $\epsilon(s(t_i))$. For instance, it generates the mean and standard deviation of a Gaussian distribution.

**Reward function learning**: We minimize the error

$$\|r(t_i) - \kappa\big(s(t_i), a_i, \xi(s(t_i)\big)\|_2^2, \tag{3}$$

where $r(t_i)$ is the true reward at timestamp $t_i$, e.g., the event exposure counts. $\xi$ is Gaussian noise and $\kappa$ is a neural network with reparameterization trick.

**Value function learning**: Let $\tau_i$ be the time interval between $t_i$ and $t_{i+1}$, i.e., $\tau_i = t_{i+1} - t_i$. In SMDPs, the target function for $V(s(t_i))$ is $\hat{V}\big(s(t_i)\big) := \frac{1-e^{-\rho\tau_i}}{\rho} r(t_i) + e^{-\rho\tau_i} V\big(s(t_{i+1})\big)$. Such target function can be derived from (1), in particular, a sampled version of Bellman equation in SMDPs. We defer derivations to Appendix. Comparing with the temporal difference learning in MDPs (Sutton, Barto et al. 1998), we have an additional stochastic term $\tau$ corresponding to the inter-event time interval. Recall that we have already obtained the NHPI model, the transition and reward model, and therefore we can generate imaginary state $s(t_{i+1})$ and reward $r(t_i)$ easily. The time interval $\tau_i$ can be simulated by NHPI with Thinning algorithm (Ogata 1981), and thus the target function defined above can be easily obtained in practice. Remark that explicitly modelling of $\tau$ with NHPI is one *key* advantage of our method over the conventional RL approaches (Haarnoja et al. 2018; Heess et al. 2015). While conventional approach can encode the timestamp using RNN naively, we can capture the stochasticity of the time interval $\tau$ much better by TPP. It is evidenced by our experiment, where the inputs of all algorithms are the same, i.e., the sequence of the observation $\{(t_i, k_i, a_i)\}_{i=1}^L$, while our method outperforms the conventional RL a lot. The term $(1 - e^{-\rho\tau_i})/\rho$ is the accumulated discount factor of the interval. We approximate the value function $V$ by a neural network $V_\phi$ and minimize the following error term w.r.t $\phi$.

$$\|V_\phi(s(t_i)) - \hat{V}_\phi(s(t_i))\|_2^2. \tag{4}$$

Please note that when we optimize above equation, the target value $\hat{V}_\phi$ is fixed, as a common practice in RL community (Haarnoja et al. 2018).

**Policy learning:** We apply the stochastic policy so that the agent can explore the environment thoroughly (Haarnoja et al. 2018). In particular, we parameterize the policy $\pi(s(t_i))$ by a neural network $\pi_\theta(s(t_i))$. If the action space is discrete, we reparameterize the output by the Gumbel-softmax trick (Jang, Gu, and Poole 2016); If it is continuous, we reparameterize the output by the Gaussian distribution (Kingma and Welling 2013). Again, we sample the Bellman equation (1) but with the learned dynamics and value function (see derivations in Appendix ) as that in SVG (Heess et al. 2015) and Dreamer (Hafner et al. 2019a). Specifically, we replace $s(t_{i+1})$ by $g(s(t_i), \pi(s(t_i)), \epsilon(s(t_i)))$, replace $r(t_i)$ by learned reward $\kappa(s(t_i), \pi(s_i), \xi(s(t_i)))$, replace action by $\pi_\theta(s(t_i))$ and rollout Bellman equation one step to obtain

$$\begin{aligned} V_{\phi,\theta}(s(t_i)) = (1 - e^{-\rho\tau_i})\kappa\big(s(t_i), \pi_\theta(s(t_i)), \xi(s(t_i))\big)/\rho \\ + e^{-\rho\tau_i} V_\phi\Big(g\big(s(t_i), \pi_\theta(s(t_i)), \epsilon(s(t_i))\big)\Big). \end{aligned} \tag{5}$$

We can see $V_{\phi,\theta}$ is a function of policy $\pi_\theta$. We do one policy improvement step as that in SVG (Heess et al. 2015),

$$\theta \leftarrow \theta + \alpha \frac{\partial V_{\phi,\theta}}{\partial \theta}, \tag{6}$$

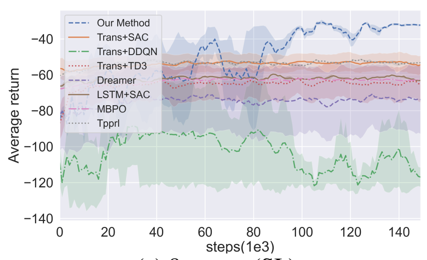
where $\alpha$ is the learning rate.

**Algorithm:** We combine all pieces discussed above and propose the stochastic event driven reinforcement learning (SEDRL) which consists of three parts: 1. The stochastic agent collects the real data trajectory from the environment, and train the NHPI by maximizing the likelihood of NHPI. 2. The agent learns the transition and reward model on $\tilde{\mathbf{h}}(t_i)$ (the state $s(t_i)$ in our algorithm). 3. The agent learns the value function $V_\phi$ and improves the policy using (6). The pseducode of the algorithm is deferred to Appendix.

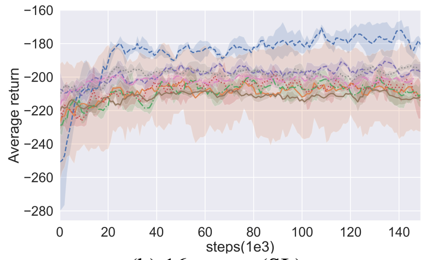
## Experiment

In this section, we evaluate our algorithm SEDRL comprehensively in the synthetic simulation and experiments with real data such as smart broadcasting and improving the engagement of the social media platform. In the real-data experiment, we follow the literature to build the simulator with real-world data (Upadhyay, De, and Gomez-Rodriguez 2018; Zarezade et al. 2017), since directly deploying RL agent in the scenarios like social media and healthcare is difficult and may cause safety and ethics issue. We compare our algorithm with multiple state-of-the-art deep RL methods including model-free algorithms such as Tpprl (Upadhyay, De, and Gomez-Rodriguez 2018), SAC (Haarnoja et al. 2018), TD3 (Fujimoto, Hoof, and Meger 2018), DDQN (Van Hasselt, Guez, and Silver 2016) and model-based ones such as Dreamer (Hafner et al. 2019a) and MBPO (Janner et al. 2019). Among them, Tpprl explicitly considers the stochasticity of the time interval $\tau$ and devices a model-free policy gradient algorithm. For the other conventional RL baselines, the problem is formulated as MDP where timestamps are just regular features for the neural networks. We conduct the ablation study on the SMDP formulation in appendix.

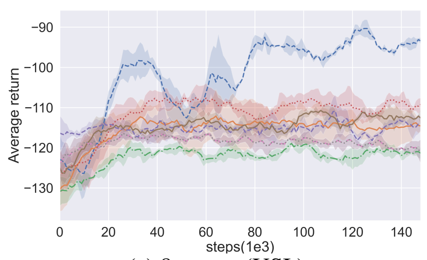
The inputs for all algorithms are the sequences of the observation, i.e., $\{(t_j, k_j, a_j)\}_{j=1,..,L}$. We equip aforementioned baselines with LSTM (Hochreiter and Schmidhuber



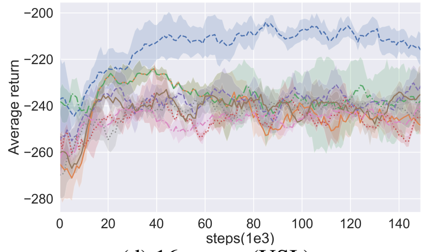
(a) 8 events (SI )



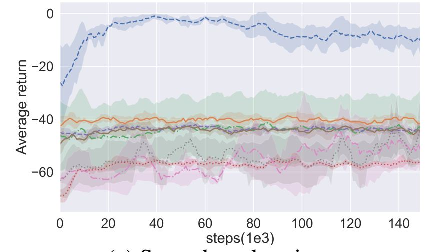
(b) 16 events (SI )



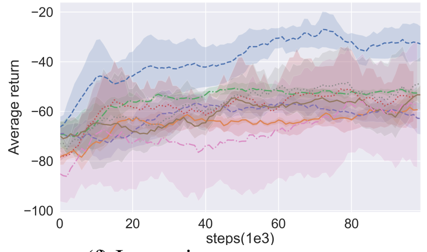
(c) 8 events (USI )



(d) 16 events (USI )



(e) Smart broadcasting



(f) Improving engagements

Figure 3: Performance of proposed methods and other baselines. The x-axis is the training step and the y-axis is the cumulative reward. Each experiment is tested on ten trials using various random seeds and initialized parameters. The solid line is the mean of the average return. The shaded region represents the standard deviation.

1997) or Transformer (Vaswani et al. 2017) in the value function network and policy network to encode the observations into state rather than simple feed-forward neural networks as usual, since the environment is highly partially observable and the observation is in the form of sequence (Oh et al. 2015). Notice that in our algorithm the hidden space is obtained by the Transformer structure (multi-head attention+MLP). Therefore the performance gap between our SEDRL and baselines are *only* due to the performance of the algorithm itself. Some details of the baselines are in the following: **1)** Tpprl: Tpprl is a model-free RL algorithm tailored for marked temporal point processes (Upadhyay, De, and Gomez-Rodriguez 2018). **2)** Transformer+SAC: SAC is a strong baselines in RL and achieves robust performance in almost all test environment (Haarnoja et al. 2018). We implement Transformer to encode the event sequence (Parisotto et al. 2020; Vaswani et al. 2017) . Specifically, we stack two multi-head attention layers with head number equal two. Then we feed this embedding as the input to discrete action SAC (Haarnoja et al. 2018; Christodoulou 2019). **3)** Transformer+DDQN: This baseline is similar to the Transformer+SAC, but we replace SAC by DDQN. **4)** Transformer+TD3: We replace SAC by discrete action version of TD3. **5)** Dreamer: Dreamer is a model-based algorithm where the deterministic transition model is modeled by LSTM. We replace the CNN in Dreamer by MLP since the input is not image. **6)** MBPO: MBPO applies the model ensemble in the dynamic model learning to enhance the performance. We replace the feed-forward neural network in MBPO by LSTM. **7)** LSTM+SAC: The sequence encoding is obtained by the LSTM. Other details such as the hyperparameter tunnings in SEDRL and baselines are in Appendix.

## Experimental Result

**Synthetic simulation:** we simulate the fake news mitigation problem (Farajtabar et al. 2017). The recent proliferation of malicious fake news in social media has been a source of widespread concern. In this simulation, the platform is the agent and the users are the environment. The social media platform hopes to optimize the performance of real news propagation. Intuitively, it wants to steer the spontaneous user mitigation activities by exposing more valid news. The action is an event that is selected from some predefined valid news(event) set, which hopefully can distract the attention of users on the rumor. We create the environment by constructing a simulator of Hawkes process with exogenous intervention (Wang et al. 2018) using Thinning algorithm (Ogata 1981). At time step $t_i$, the input of the simulator is the action $a_i$ taken by the agent. The $k$-th action type is encoded as a one-hot vector where $k$-th entry is one. Zero vector means no action taken. The simulator gets the action as an input and generates the event $v_{i+1} \in \{1, ..., K\}$ at $t_{i+1}$ ($K$=8 or 16 in our simulation). The parameter such as $\beta_{ij}$ in the simulator are some predefined values. The goal of the platform is to find the best policy that can control the event (valid and fake news) intensity vector such that it is close to a predefined level $\boldsymbol{\lambda}_{target}$. Hence, the reward of our simulation for each time step $r_t = -\|\boldsymbol{\lambda}_{target} - \boldsymbol{\lambda}_t\|^2 - 0.1 * \text{cost}(a_t)$ provided by the simulator, where $\boldsymbol{\lambda}_t$ is the intensity at $t$. Cost of action is 1 if the agent inserts an event and otherwise 0 if we do nothing.

Two intervention settings are considered in our simulation, i.e., synchronized intervention setting and unsynchronized intervention setting. We use *SI* and *USI* for short respectively. Notice that the events occur asynchronously in both settings. The SI setting is straightforward, with the aim to show that even if in an easy setting, the baselines can not learn a good policy. Specifically, in the SI setting, the agent can intervene in the sequence at evenly discretized time interval, i.e., at timestamp $\{\Delta t, 2\Delta t, .., i\Delta, ...\}$ (see the illustration in Appendix of experiment setting. This setting is simpler than our original setting (USI) in the section of problem setup, where the intervention can be applied at $\{t_1, t_2, ..., t_i, ..\}$, i.e., at the timestamps of event $i$. The simulation is terminated when time $t$ exceeds the maximum time horizon (e.g., 100 time units.). In general, in this setting, the agent has more chances to intervene with the environment (in our simulation $\Delta < (t_{i+1} - t_i)$). In addition, since time is evenly discretized, this setting may be more friendly to the baselines. Also note that our algorithm (and baselines) can easily adapt to this setting, since our model can easily accommodate by setting an additional null event occurring at timestamp $i\Delta$. The USI setting is our setting in the problem setup section, which means the agent only makes decision when a specific event happens along the trajectory (see Figure 1), which is more practical in real-world problems. For instance, the doctor would change the treatment only after the patient's visit. In our simulation, the agent inserts actions at the next coming clock tick following the occurrence of events. The simulation is terminated when the time $t$ exceeds the maximum time horizon. In both settings, we conduct two simulations with K=8 and K=16 events respectively. We use half of these events as our action set (i.e., 4 events and 8 events respectively). Details of the synthetic simulation are deferred to the appendix of experiment setting. All experiments in Figure 3 are repeated with ten trials with various random seeds and initial parameters.

Figure 3 (a) and (b) demonstrate the results in the SI setting. In (a), our method SEDRL has the best result followed by Tpprl, Transformer+SAC, LSTM+ SAC, MBPO, Transformer+TD3, Dreamer and then Transformer+DDQN (Trans+Q). In (b), our SEDRL also achieves the best result. The runner up is Dreamer, while other baselines have similar results around -210. Figure 3 (c) and (d) present the results in the USI setting. We can derive similar conclusion as that in the SI setting. Our method outperforms conventional RL algorithm since we consider the dynamics that can be interrupted by the stochastic event. Comparing with Tpprl, our method explicitly embeds the dynamic model into the Bellman equation of SMDPs and therefore it is much more sample efficient than the model-free policy gradient algorithm.

**Smart broadcasting:** In this problem, the user of the social media wants to decide what to post to elicit the attention from her followers (Spasojevic et al. 2015). The user herself is the agent and she generates action event (post tweets). The environment is formed by her followers and it generates feedback events if her followers retweet. Since we cannot make real interventions on the platform of twitter , we follow the conventions in the literature (Upadhyay, De, and Gomez-Rodriguez 2018; Zarezade et al. 2017) to train a simulator with real-world data. In particular, we use Retweet dataset (Zhao et al. 2015) to learn a model of follower. The Retweets dataset contains sequences of tweets, where each sequence contains origin tweets (i.e., some user initiates a tweet), and some follow-up tweets. We group the post and retweet into three categories event (i.e., $K$=3) following the work in (Zuo et al. 2020). Then follower's behavior is learned by a neural temporal point process (Zuo et al. 2020) with exogenous excitation (i.e., the origin tweets) (Ding 2020). The users should expose the predefined events (action set are three kinds of post) to her followers to elicit attentions and then observe the feedback event of the followers. Each kind of the feedback event (retweet) and action event (post) corresponds to a predefined reward and cost respectively( appendix G), where the aim of action cost is to discourage too much intervention over the environment. The goal of the agent is to maximize the total reward minus the cost over the simulation time [0,T]. Given the feedback model of the follower (generated by the simulator), we can do simulations through the interactions over the environment. Some details of the experiment setting are deferred to the appendix . In Figure 3 (e), we demonstrate the performance of all algorithm. Our method reaches asymptotic result after only 20k training steps and achieves $-10$ long-term reward, while the performances of baselines are below $-40$.

**Improving the Engagement:** The platforms of e-commerce or social media sometimes award the users to promote engagement in the community. In this problem, we would design a policy to help the platform to hand out awards. Take Stack-Overflow for instance, where users can use post to ask questions or seek advises in various technical areas while the other users can answer the questions freely and the answer will be rated or awarded by other views. Under our problem setting, the agent is the platform, while the environment consist of the users in the website. The action event correspond to badge types, e.g., Nice Question, Good Answer, etc, and the event times correspond to the award times. The platform can encourages user engagement by giving certain types of badges (action) to the user after he receives some other badge. For example, when a user receives "Good Answer" badge, the agent awards a "Great Answer" or "Guru" badge (as an action) to encourage his/her engagement in the website. Then the platform can observe the feedback event from the users. For instance, The user may answer questions more frequently with higher quality (e.g., more useful answers). Each feedback event and action event have a corresponding predefined reward or cost. We sum all of them over a certain period to reflect the engagement rate of the platform. The agent should learn a control policy to maximize the long-term reward by manipulating the action events. Since we can not deploy the policy in the Stack-Overflow, we use the data gathered from Stack-Overflow (Leskovec and Krevl 2014) to learn a feedback model of the users. The data is collected in a two-year period, and we treat each user's reward history as a sequence following the work (Zuo et al. 2020). We pick eleven types of the event in the dataset as the exogenous excitation while the other half as the endogenous one. Therefore, the size of the action set is eleven and $K = 22$ (See more details in appendix). We can learn a user feedback model using neural temporal process with exogenous excitation (Ding 2020). Using this simulator (feedback model) trained from the real data set, we can do experiments through the interactions with the environment. We report results in Figure 3 (f). Our method could achieve $-35$ cumulative reward after 100k step training, while the rewards of baselines are below $-50$.

## References

Bradtke, S. J.; and Duff, M. O. 1995. Reinforcement learning methods for continuous-time Markov decision problems. In *Advances in neural information processing systems*, 393–400.

Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; and Zaremba, W. 2016. Openai gym. *arXiv preprint arXiv:1606.01540*.

Christodoulou, P. 2019. Soft actor-critic for discrete action settings. *arXiv preprint arXiv:1910.07207*.

Cox, D. R.; and Isham, V. 1980. *Point processes*, volume 12. CRC Press.

Dalal, G.; Gilboa, E.; and Mannor, S. 2016. Hierarchical decision making in electricity grid management. In *International Conference on Machine Learning*, 2197–2206.

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

Ding, R. 2020. *TRAFFIC CONGESTION MODELING WITH DEEP ATTENTION HAWKES PROCESS*. Ph.D. thesis, Georgia Institute of Technology.

Du, J.; Futoma, J.; and Doshi-Velez, F. 2020. Model-based reinforcement learning for semi-markov decision processes with neural odes. *Neurips2020*.

Du, N.; Dai, H.; Trivedi, R.; Upadhyay, U.; Gomez-Rodriguez, M.; and Song, L. 2016. 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*, 1555–1564.

Farajtabar, M.; Yang, J.; Ye, X.; Xu, H.; Trivedi, R.; Khalil, E.; Li, S.; Song, L.; and Zha, H. 2017. Fake news mitigation via point process based intervention. In *International Conference on Machine Learning*, 1097–1106. PMLR.

Fujimoto, S.; Hoof, H.; and Meger, D. 2018. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, 1587–1596. PMLR.

Gu, S.; Holly, E.; Lillicrap, T.; and Levine, S. 2017. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *2017 IEEE international conference on robotics and automation (ICRA)*, 3389–3396. IEEE.

Ha, D.; and Schmidhuber, J. 2018. World models. *arXiv preprint arXiv:1803.10122*.

Haarnoja, T.; Zhou, A.; Abbeel, P.; and Levine, S. 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, 1861–1870. PMLR.

Hafner, D.; Lillicrap, T.; Ba, J.; and Norouzi, M. 2019a. Dream to control: Learning behaviors by latent imagination. *ICLR2020*.

Hafner, D.; Lillicrap, T.; Fischer, I.; Villegas, R.; Ha, D.; Lee, H.; and Davidson, J. 2019b. Learning latent dynamics for planning from pixels. In *International Conference on Machine Learning*, 2555–2565. PMLR.

Hawkes, A. G.; and Oakes, D. 1974. A cluster process representation of a self-exciting process. *Journal of Applied Probability*, 493–503.

Heess, N.; Wayne, G.; Silver, D.; Lillicrap, T.; Tassa, Y.; and Erez, T. 2015. Learning continuous control policies by stochastic value gradients. *arXiv preprint arXiv:1510.09142*.

Hochreiter, S.; and Schmidhuber, J. 1997. Long short-term memory. *Neural computation*, 9(8): 1735–1780.

Jang, E.; Gu, S.; and Poole, B. 2016. Categorical reparameterization with gumbel-softmax. *ICLR2017*.

Janner, M.; Fu, J.; Zhang, M.; and Levine, S. 2019. When to trust your model: Model-based policy optimization. *Neurips2019*.

Kingma, D. P.; and Welling, M. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Lazic, N.; Boutilier, C.; Lu, T.; Wong, E.; Roy, B.; Ryu, M.; and Imwalle, G. 2018. Data center cooling using model-predictive control. In *Advances in Neural Information Processing Systems*, 3814–3823.

Leskovec, J.; and Krevl, A. 2014. SNAP Datasets: Stanford Large Network Dataset Collection. http://snap.stanford.edu/data. Accessed: 2022-09-01.

Lewis, P. W.; and Shedler, G. S. 1979. Simulation of nonhomogeneous Poisson processes by thinning. *Naval research logistics quarterly*, 26(3): 403–413.

Mei, H.; and Eisner, J. M. 2017. The neural hawkes process: A neurally self-modulating multivariate point process. *Advances in Neural Information Processing Systems*, 30: 6754–6764.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. *nature*, 518(7540): 529–533.

Munos, R. 2006. Policy gradient in continuous time. *Journal of Machine Learning Research*, 7(May): 771–791.

Ogata, Y. 1981. On Lewis' simulation method for point processes. *IEEE transactions on information theory*, 27(1): 23–31.

Oh, J.; Guo, X.; Lee, H.; Lewis, R. L.; and Singh, S. 2015. Action-conditional video prediction using deep networks in atari games. *Advances in neural information processing systems*, 28: 2863–2871.

Parisotto, E.; Song, F.; Rae, J.; Pascanu, R.; Gulcehre, C.; Jayakumar, S.; Jaderberg, M.; Kaufman, R. L.; Clark, A.; Noury, S.; et al. 2020. Stabilizing transformers for reinforcement learning. In *International Conference on Machine Learning*, 7487–7498. PMLR.

Rasmussen, J. G. 2018. Lecture notes: Temporal point processes and the conditional intensity function. *arXiv preprint arXiv:1806.00221*.

Rizoiu, M.-A.; Lee, Y.; Mishra, S.; and Xie, L. 2017. A tutorial on hawkes processes for events in social media. *arXiv preprint arXiv:1708.06401*.

Shalev-Shwartz, S.; Shammah, S.; and Shashua, A. 2016. Safe, multi-agent, reinforcement learning for autonomous driving. *arXiv preprint arXiv:1610.03295*.

Spasojevic, N.; Li, Z.; Rao, A.; and Bhattacharyya, P. 2015. When-to-post on social networks. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2127–2136.

Sutton, R. S.; Barto, A. G.; et al. 1998. *Introduction to reinforcement learning*, volume 135. MIT press Cambridge.

Sutton, R. S.; Precup, D.; and Singh, S. 1999. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*.

Upadhyay, U.; De, A.; and Gomez-Rodriguez, M. 2018. Deep reinforcement learning of marked temporal point processes. *Neurips 2018*.

Van Hasselt, H.; Guez, A.; and Silver, D. 2016. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In *Advances in neural information processing systems*, 5998–6008.

Wang, Y.; Theodorou, E.; Verma, A.; and Song, L. 2018. A stochastic differential equation framework for guiding online user activities in closed loop. In *International Conference on Artificial Intelligence and Statistics*, 1077–1086. PMLR.

Xue, S.; Shi, X.; Hao, H.; Ma, L.; Zhang, J.; Wang, S.; and Wang, S. 2021. A Graph Regularized Point Process Model For Event Propagation Sequence. In *2021 International Joint Conference on Neural Networks (IJCNN)*, 1–7.

Xue, S.; Shi, X.; Zhang, Y. J.; and Mei, H. 2022. HYPRO: A Hybridly Normalized Probabilistic Model for Long-Horizon Prediction of Event Sequences. In *Advances in Neural Information Processing Systems*.

Yıldız, Ç.; Heinonen, M.; and Lähdesmäki, H. 2021. Continuous-Time Model-Based Reinforcement Learning. *ICML2021*.

Zarezade, A.; De, A.; Upadhyay, U.; Rabiee, H. R.; and Gomez-Rodriguez, M. 2017. Steering Social Activity: A Stochastic Optimal Control Point Of View. *J. Mach. Learn. Res.*, 18: 205–1.

Zhang, Q.; Lipani, A.; Kirnap, O.; and Yilmaz, E. 2020. Self-attentive hawkes process. In *International Conference on Machine Learning*, 11183–11193. PMLR.

Zhao, Q.; Erdogdu, M. A.; He, H. Y.; Rajaraman, A.; and Leskovec, J. 2015. 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*, 1513–1522.

Zuo, S.; Jiang, H.; Li, Z.; Zhao, T.; and Zha, H. 2020. Transformer Hawkes Process. *ICML 2020*.