Conditional Generative Modeling for High-dimensional Marked Temporal Point Processes

Anonymous Author(s)
Affiliation
Address
`email`

Abstract

Recent advancements in generative modeling have made it possible to generate high-quality content from context information, but a key question remains: how to teach models to know when to generate content? To answer this question, this study proposes a novel event generative model that draws its statistical intuition from marked temporal point processes, and offers a clean, flexible, and computationally efficient solution for a wide range of applications involving the generation of asynchronous events with high-dimensional marks. We use a conditional generator that takes the history of events as input and generates the high-quality subsequent event that is likely to occur given the prior observations. The proposed framework offers a host of benefits, including considerable representational power to capture intricate dynamics in multi- or even high-dimensional event space, as well as exceptional efficiency in learning the model and generating samples. Our numerical results demonstrate superior performance compared to other state-of-the-art baselines.

1 Introduction

Generating future events is a challenging yet fascinating task, with numerous practical applications [2, 9, 16, 31]. For instance, a news agency may need to generate news articles in a timely manner, taking into account the latest events and trends. Similarly, an online shopping platform may aim to provide highly personalized recommendations for products, services, or content based on a user's preferences and behavior patterns over time, as shown in Figure 1. These types of applications are ubiquitous in daily life, and the related data typically consist of a sequence of events that denote when and where each event occurred, along with additional descriptive information such as category, volume, and even text or image, commonly referred to as "marks". Recent improvements in generative modeling have made it possible to generate high-quality content from contextual information such as language descriptions. However, it remains an open question: how to teach these models to determine the appropriate timing for generating such content based on the history of events.

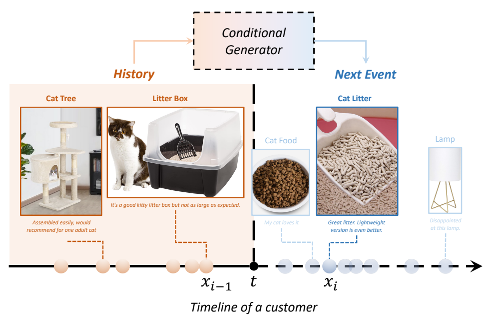


Figure 1: An example of generating high-dimensional content over time. In this example, the conditional generator explores the customer's next possible activity, including not only the purchase time, but also the item, and even its image or review. The observed events from the customer's past purchases are represented by yellow dots, while the next generated event is indicated by a blue dot.

Point processes have been a popular tool for modeling and generating asynchronous and discrete event data. With the rise of complex systems, advanced neural point processes [6, 18, 25] are proposed as powerful methods to model and simulate data by capturing complex dependencies among observed events. However, due to the use of neural networks, the model likelihood is often analytically intractable, requiring complex and expensive approximations during learning. More seriously, these

Submitted to 37th Conference on Neural Information Processing Systems (NeurIPS 2023). Do not distribute.

models face significant limitations in *generating events with high-dimensional marked information*, as the event simulation relies heavily on the thinning algorithm [20], which can be costly or even impossible when the mark space is high-dimensional. This significantly restricts the applicability of these models to modern applications [30, 34], where event data often come with high-dimensional marks, such as texts and images in police crime reports or social media posts.

To tackle these challenges, this paper introduces a novel combination of generative framework and marked temporal point processes for efficient modeling and generation of high-quality asynchronous events with high-dimensional marks. The effectiveness of our model is rooted in the ability to approximate the underlying high-dimensional data distribution through generated samples by a conditional generator, which takes the history of events as its input. The event history is summarized by a recurrent neural architecture, allowing for flexible selection based on the application's needs. The benefits of our model can be summarized by:

1. Our model is capable of handling time-stamped high-dimensional marks such as images or texts, leveraging the power of generative models within the framework of marked point processes;
2. Our model possesses superior representative power, as it does not confine the conditional intensity or probability density of the events to any specific parametric form;
3. Our model outperforms existing state-of-the-art baselines in terms of estimation accuracy and generating high-quality event series;
4. Our model excels in computational efficiency during both the training phase and the event generation process. In particular, our method needs only $\mathcal{O}(N_T)$ for generating N_T events, in contrast to the thinning algorithm's complexity of $\mathcal{O}(N^d \cdot N_T)$, where $N \gg N_T$ and d represents the event dimension.

It is important to note that our proposed framework is general and model-agnostic, meaning that a wide spectrum of generative models and learning algorithms can be applied within our framework. We present two possible learning algorithms in the Appendix A.

2 Methodology

2.1 Background: Marked temporal point processes

Marked temporal point processes (MTPPs) [23] consist of a sequence of *discrete events* over time. Each event is associated with a (possibly multi-dimensional) *mark* that contains detailed information of the event. Let $T > 0$ be a fixed time-horizon, and $\mathcal{M} \subseteq \mathbb{R}^d$ be the space of marks. We denote the space of observation as $\mathcal{X} = [0, T) \times \mathcal{M}$ and a data point in the discrete event sequence as

$$x = (t, m), \quad t \in [0, T), \quad m \in \mathcal{M},$$

where t is the event time and m represents the mark. Let N_t be the number of events up to time $t < T$ (which is random), and $\mathcal{H}_t := \{x_1, x_2, \ldots, x_{N_t}\}$ denote historical events. Let $\mathbb{N}$ be the counting measure on $\mathcal{X}$, *i.e.*, for any measurable $S \subseteq \mathcal{X}$, $\mathbb{N}(S) = |\mathcal{H}_t \cap S|$. For any function $\phi : \mathcal{X} \to \mathbb{R}$, the integral with respect to the counting measure is defined as $\int_S \phi(x) d\mathbb{N}(x) = \sum_{x_i \in \mathcal{H}_T \cap S} \phi(x_i)$. The events' distribution in MTPPs can be characterized via the conditional intensity function λ, which is defined to be the occurrence rate of events in the marked temporal space $\mathcal{X}$ given the events' history $\mathcal{H}_{t(x)}$, *i.e.*,

$$\lambda(x|\mathcal{H}_{t(x)}) = \mathbb{E}\left(d\mathbb{N}(x)|\mathcal{H}_{t(x)}\right)/dx, \tag{1}$$

where $t(x)$ extracts the occurrence time of the possible event x. Given the conditional intensity function λ, the corresponding conditional probability density function (PDF) can be written as

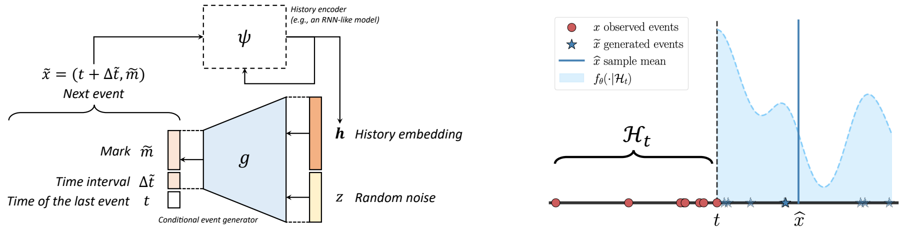
$$f(x|\mathcal{H}_{t(x)}) = \lambda(x|\mathcal{H}_{t(x)}) \cdot \exp\left(-\int_{[t_n, t(x)) \times \mathcal{M}} \lambda(u|\mathcal{H}_{t(u)}) du\right). \tag{2}$$

where t_n denotes the time of the most recent event before time $t(x)$. The point process models can be learned using maximum likelihood estimation (MLE). See all the derivations in Appendix B.

2.2 Conditional event generator

The main idea of the proposed framework is to use a *conditional event generator* to produce the i-th event $x_i = (t_{i-1} + \Delta t_i, m_i)$ given its previous $i - 1$ events. Here, Δt_i and m_i indicate the time interval between the i-th event and its preceding event and the mark of the i-th event, respectively. Formally, this is achieved by a generator function:

$$g(z, \boldsymbol{h}_{i-1}) : \mathbb{R}^{r+p} \to (0, +\infty) \times \mathcal{M},$$



(a) Model architecture

(b) Example of generated events

Figure 2: (a) The architecture of the proposed framework, which consists of two key components: A conditional generative model g that generates $(\Delta\widetilde{t}, \widetilde{m})$ given its history embedding and an RNN-like model ψ that summarizes the events in the history. (b) An example of generated one-dimensional (time only) events $\{\widetilde{x}^{(j)}\}$ given the history $\mathcal{H}_t$. The shaded area suggests the underlying conditional probability density captured by the model with parameters θ.

which takes an input in the form of a random noise vector ($z \in \mathbb{R}^r \sim \mathcal{N}(0, I)$) and a hidden embedding ($\boldsymbol{h}_{i-1} \in \mathbb{R}^p$) that summarizes the history information up to and excluding the i-th event, namely, $\mathcal{H}_{t_i} = \{x_1, \dots, x_{i-1}\}$. The output of the generator is the concatenation of the time interval and mark of the i-th event denoted by $\Delta\widetilde{t}_i$ and $\widetilde{m}_i$, respectively. To ensure that the time interval is positive, we restrict $\Delta\widetilde{t}_i$ to be greater than zero.

To represent the conditioning variable $\boldsymbol{h}_{i-1}$, we use a *history encoder* represented by ψ, which has a recursive structure such as recurrent neural networks (RNNs) [32] or Transformers [28]. In our numerical results, we opt for long short-term memory (LSTM) [7], which takes the current event x_i and the preceding hidden embedding $\boldsymbol{h}_{i-1}$ as input and generates the new hidden embedding $\boldsymbol{h}_i$. This new hidden embedding represents an updated summary of the past events including x_i. Formally,

$$\boldsymbol{h}_0 = \boldsymbol{0} \text{ and } \boldsymbol{h}_i = \psi(x_i, \boldsymbol{h}_{i-1}), \quad i = 1, 2, \dots, N_T.$$

We denote the parameters of both g and ψ using $\theta \in \Theta$. Figure 2 (a) presents the model architecture.

Connection to marked temporal point processes The proposed framework draws its statistical inspiration from MTPPs. Unlike other recent attempts at modeling point processes, our framework *approximates the conditional probability of events using generated samples* rather than directly specifying the conditional intensity in (1) or PDF in (2) using a parametric model [6, 18, 22, 24, 33].

As illustrated by Figure 2 (b), when our model generates an event denoted by $\widetilde{x} = (t + \Delta\widetilde{t}, \widetilde{m})$, it implies that the resulting event $\widetilde{x}$ follows a conditional probabilistic distribution that is determined by the model parameter θ and the event's history $\mathcal{H}_t$:

$$\widetilde{x} \sim f_\theta(x|\mathcal{H}_{t(x)}),$$

where f_θ denotes the conditional PDF of the underlying MTPP (2). This design has three main advantages compared to other point process models:

1. *Generative efficiency*: The generative nature of our model confers an exceptional efficiency in simulating a complete event series for any point processes without relying on thinning algorithms [20]. To exemplify, thinning algorithm (Algorithm 4) has a time complexity of $\mathcal{O}(N^d \cdot N_T)$ to generate N_T events from a history-dependent point process in d-dimensional space $\mathcal{X}$, with $N \gg N_T$ being the number of uniformly sampled candidates in one dimension. In contrast, our generation process (Algorithm 1) only requires a complexity of $\mathcal{O}(N_T)$.
2. *Expressiveness*: The proposed model enjoys considerable representational power, as it does not impose any restrictions on the parametric form of the conditional intensity λ or PDF f. The numerical findings also indicate that our model is capable of capturing complex event interactions, even in a multi-dimensional space.
3. *Predictive efficiency*: To predict the next event $\widehat{x}_i = (t_{i-1} + \Delta\widehat{t}_i, \widehat{m}_i)$ given the observed events' history $\mathcal{H}_{t_i}$, we can calculate the sample average over a set of generated events $\{\widetilde{x}_i^{(l)}\}$ without the need for an explicit expectation computation, *i.e.*,

$$\widehat{x}_i = \int_{(t_{i-1}, +\infty) \times \mathcal{M}} x \cdot f_\theta(x|\mathcal{H}_{t(x)}) dx \approx \frac{1}{L} \sum_{l=1}^{L} \widetilde{x}_i^{(l)},$$

where L denotes the number of samples.

Algorithm 1 Event generation process using `CEG`

Input: Generator g, history encoder ψ, time horizon T
Initialization: $\mathcal{H}_T = \emptyset, \boldsymbol{h}_0 = \boldsymbol{0}, t = 0, i = 0$
while $t < T$ **do**
  1. Sample $z \sim \mathcal{N}(0, I)$;
  2. Generate next event $\widetilde{x} = (t + \Delta\widetilde{t}, \widetilde{m})$, where $(\Delta\widetilde{t}, \widetilde{m}) = g(z, \boldsymbol{h}_i)$;
  3. $i = i + 1; t = t + \Delta\widetilde{t}; x_i = \widetilde{x}; \mathcal{H}_T = \mathcal{H}_T \cup \{x_i\}$;
  4. Update hidden embedding $\boldsymbol{h}_i = \psi(x_i, \boldsymbol{h}_{i-1})$;
end while
if $t(x_i) \geq T$ **then**
  return $\mathcal{H}_T - \{x_i\}$
else
  return $\mathcal{H}_T$
end if

3 Experiments

We evaluate our method using both synthetic and real data and demonstrate the superior performance compared to five state-of-the-art approaches, including (1) Recurrent marked temporal point processes (`RMTPP`) [6], (2) Neural Hawkes (`NH`) [18], (3) Fully neural network based model (`FullyNN`) [22], (4) Epidemic type aftershock sequence (`ETAS`) [21] model, (5) Deep non-stationary kernel in point process (`DNSK`) [5]. The first three baselines leverage neural networks to model temporal event data (or only with categorical marks). The last two baselines are chosen for testing multi-dimensional event data. Meanwhile, the `DNSK` is the state-of-the-art method that uses neural networks for high-dimensional mark modeling. In the following, we refer to our proposed method as the conditional event generator (`CEG`). Detailed experimental setup and model architectures are presented in Appendix F.

3.1 Synthetic data

We first evaluate our model on synthetic data. To be specific, we generate four one-dimensional (1D) and a three-dimensional (3D) synthetic data sets: Four 1D (time only) data sets include 1,000 sequences each, with an average length of 135 events per sequence, and are simulated by two self-exciting processes and two self-correcting processes, respectively, using thinning algorithm (Algorithm 4 in Appendix F). The 3D (time and space) data set also includes 1,000 sequences, each with an average length of 150, generated by a randomly initialized `CEG` using Algorithm 1.

To assess the effectiveness of our model in acquiring the underlying data distribution, we computed the mean relative error (MRE) of the estimated conditional intensity and PDF on the testing set, and compared them to the ground truth. Table 1 presents more quantitative results on 1D and 3D data sets, including log-likelihood testing per events and the mean relative error (MRE) of the recovered conditional density and intensity. These results demonstrate the consistent superiority of `CEG` over other methods across all scenarios. Figure F3 and Figure F4 in Appendix F presents visualizations of the estimated conditional probability density on 1D and 3D synthetic data sets, where `CEG` accurately captures the complex spatio-temporal point patterns while other baselines fail to do so.

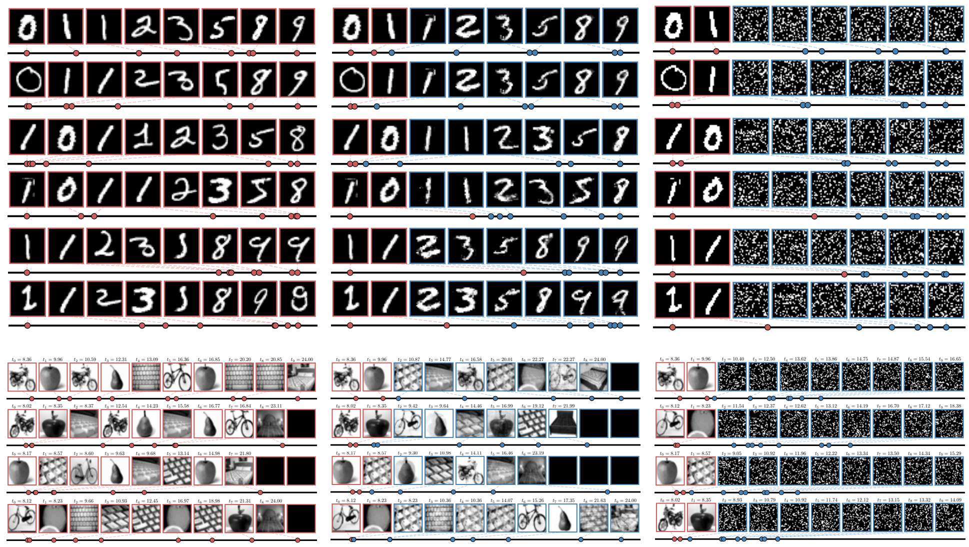
3.2 Semi-synthetic data with image marks

We test our model's capability of generating complex high-dimensional marked events on two semi-synthetic data, including time-stamped MNIST (T-MNIST) and CIFAR-100 (T-CIFAR). In these data sets, both the mark (the image category) and the timestamp are generated through a marked point process. Images from MNIST and CIFAR-100 are subsequently chosen at random based on these marks, acting as an high-dimensional representation of the original image category. It's important to note that during the training phase, categorical marks are excluded, retaining only the high-dimensional images for model learning. Since calculating the log-likelihood for event series with

Table 1: Performance comparison with five baseline methods.

	1D self-exciting data			1D self-correcting data			3D synthetic data			3D earthquake data
Model	Testing ℓ	MRE of f	MRE of λ	Testing ℓ	MRE of f	MRE of λ	Testing ℓ	MRE of f	MRE of λ	Testing ℓ
`RMTPP`	-1.051 (0.015)	0.437	0.447	-0.975 (0.006)	0.308	0.391	/	/	/	/
`NH`	-0.776 (0.035)	0.175	0.198	-1.004 (0.010)	0.260	0.363	/	/	/	/
`FullyNN`	-1.025 (0.003)	0.233	0.330	-0.821 (0.008)	0.322	0.495	/	/	/	/
`ETAS`	/	/	/	/	/	/	-4.832 (0.002)	0.981	0.902	-3.939 (0.002)
`DNSK`	-0.649 (0.002)	0.015	**0.024**	-2.832 (0.004)	0.134	0.185	-2.560 (0.004)	0.339	0.415	-3.606 (0.003)
`CEG`	**–0.645** (0.002)	**0.013**	0.066	**–0.768** (0.005)	**0.042**	**0.075**	**–2.540** (0.011)	**0.049**	**0.089**	**–2.629** (0.015)

*Numbers in parentheses present standard error for three independent runs.



(a) True series (b) CEG generated series (c) DNSK generated series

Figure 3: Generated T-MNIST (first row) and T-CIFAR (second row) series using CEG and a neural point process baseline DNSK, with true sequences displayed in the first column. Each event series is generated (blue boxes) given the first two true events (red boxes).

high-dimensional marks is infeasible for CEG (the number of samples needed to estimate density is impractically large), we evaluate the model performance according to: (1) the quality of the generated image marks and (2) the transition dynamics of the entire series. Details of the data generation processes can be found in Appendix F.

1. T-MNIST: For each sequence in the data, the actual digit in the succeeding image is the aggregate of the digits in the two preceding marks. The initial two digits are randomly selected from 0 and 1. The digits in the marks must be less than nine. The hand-written image for each mark is then chosen from the corresponding subset of MNIST according to the digit. The time for the entire MNIST series conforms to a Hawkes process with an exponentially decaying kernel.
2. T-CIFAR: The data contains event series that depict a typical day in the life of a graduate student, spanning from 8:00 to 24:00. The marks are sampled from four categories: outdoor exercises, food ingestion, working, and sleeping. Depending on the most recent activity, the subsequent one is determined by a transition probability matrix. Images are selected from the respective categories to symbolize each activity. The activity times follows a self-correcting process.

Figure 3 presents the true T-MNIST series alongside the series generated by CEG and DNSK given the first two events. Our model not only generates high-dimensional event marks that resemble true images, but also successfully captures the underlying data dynamics, such as the clustering patterns of the self-exciting process and the transition pattern of image marks. On the contrary, the DNSK only learns the temporal effects of historical events and struggles to estimate the conditional intensity for the high-dimensional marks. Besides, the grainy images generated by DNSK demonstrate the challenge of simulating credible high-dimensional content using thinning algorithm. This is because the real data points, being sparsely scattered in the high-dimensional mark space, make it challenging for the candidate points to align closely with them in the thinning algorithm.

Similar results are shown in Figure 3 on T-CIFAR data set, where the CEG is able to simulate high-quality daily activities with high-dimensional content at appropriate times. However, the DNSK fails to extract any meaningful patterns from the data, since intensity-based modeling and data generation become ineffectual in high-dimensional mark space.

3.3 Real data

In our real data results, our model demonstrates superior efficacy in generating multi- and high-dimensional event sequences of high quality, which closely resemble real event series.

Northern California earthquake catalog We test our method using the Northern California Earthquake Data [19], which contains detailed information on the timing and location of earthquakes that occurred in central and northern California from 1978 to 2018, totaling 5,984 records with

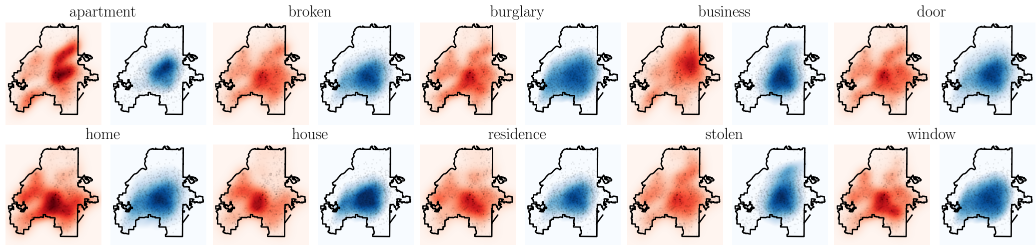
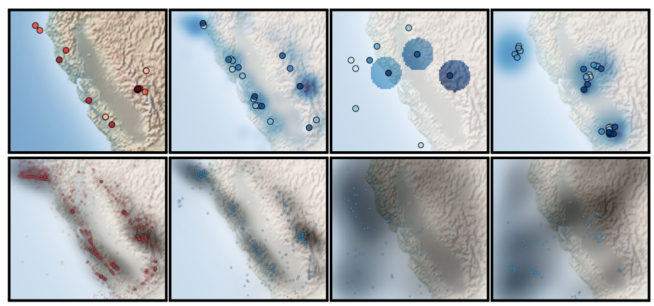


Figure 5: The spatial distributions of the TF-IDF values of 10 crime-related keywords. The heatmap in red and blue represent distributions of TF-IDF value of the keywords in the true and generated events, respectively. The black dots pinpoint the locations of the corresponding events.

magnitude greater than 3.5. We divided the data into several sequences by month. In comparison to other baseline methods that can only handle 1D event data, we primarily evaluated our model against `DNSK` and `ETAS`. we assess the quality of the generated sequences by each model. Our model's generation process for new sequences can be efficiently carried out using Algorithm 1, whereas both `DNSK` and `ETAS` requires the use of a thinning algorithm (Algorithm 4) for simulation. We also compared the estimated conditional probability density functions (PDFs) of real sequences by each model in Appendix F.



(a) Real (b) `CEG` (c) `DNSK` (d) `ETAS`

Figure 4: Comparison between real and generated earthquake sequence. The first row displays a single sequence, either real or generated, with the color depth of the dots reflecting the occurrence time of each event. Darker colors represent more recent events. The shaded areas represent the estimated conditional PDFs. The second row shows 1,000 real or generated events, where the gray area indicates the high density of events, which can be interpreted as the "background rate".

We compare the generative ability of each method in Figure 4. The top left sub-figure features a single event series selected at random from the data set, while the rest of the sub-figures in the first row exhibit event series generated by each model, respectively. The quality of the generated earthquake sequence using our method is markedly superior to that generated by `DNSK` and `ETAS`. We also simulate multiple sequences using each method and visualize the spatial distribution of generated earthquakes in the second row. The shaded area reflects the spatial density of earthquakes obtained by KDE and represents the "background rate" over space. It is evident that `CEG` is successful in capturing the underlying earthquake distribution, while the two STPP baselines are unable to do so. Additional results in Figure F6 visualizes the conditional PDF estimated by `CEG`, `DNSK`, and `ETAS` for an actual earthquake sequence in testing set, respectively. The results indicate that our model is able to capture the heterogeneous triggering effects among earthquakes which align with current understandings of the San Andreas Fault System [29]. However, both `DNSK` and `ETAS` fail to extract this geographical feature from the data.

Atlanta crime reports with textual description We further assess our method using 911-calls-for-service data in Atlanta. The proprietary data set contains 4644 burglary incidents from 2016 to 2017, detailing the time, location, and a comprehensive textual description of each incident. Each textual description was transformed into a TF-IDF vector [1], from which the top 10 keywords with the most significant TF-IDF values were selected. The location combined with the corresponding 10-dimensional TF-IDF vector is regarded as the mark of the incident. We first fit our `CEG` model using the preprocessed data, subsequently generate crime event sequences, and then compare them with the real data.

Figure 5 visualizes the spatial distributions of the true and the generated TF-IDF value of each keyword, respectively, signifying the heterogeneous crime patterns across the city. As we can observe, our model is capable of capturing such spatial heterogeneity for different keywords and simulating crime incidents that follow the underlying spatio-temporal-textual dynamics existing in criminological *modus operandi* [34].

References

[1] Akiko Aizawa. An information-theoretic perspective of tf–idf measures. *Information Processing & Management*, 39(1):45–65, 2003.

[2] Homanga Bharadhwaj, Homin Park, and Brian Y Lim. Recgan: recurrent generative adversarial networks for recommendation systems. In *Proceedings of the 12th ACM Conference on Recommender Systems*, pages 372–376, 2018.

[3] Sam Bond-Taylor, Adam Leach, Yang Long, and Chris G Willcocks. Deep generative modelling: A comparative review of vaes, gans, normalizing flows, energy-based and autoregressive models. *IEEE transactions on pattern analysis and machine intelligence*, 2021.

[4] Xiuyuan Cheng and Hau-Tieng Wu. Convergence of graph laplacian with knn self-tuned kernels. *Information and Inference: A Journal of the IMA*, 11(3):889–957, 2022.

[5] Zheng Dong, Xiuyuan Cheng, and Yao Xie. Spatio-temporal point processes with deep non-stationary kernels. *arXiv preprint arXiv:2211.11179*, 2022.

[6] Nan Du, Hanjun Dai, Rakshit Trivedi, Utkarsh Upadhyay, Manuel Gomez-Rodriguez, and Le 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*, pages 1555–1564, 2016.

[7] Alex Graves and Alex Graves. Long short-term memory. *Supervised sequence labelling with recurrent neural networks*, pages 37–45, 2012.

[8] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

[9] Ren-Hung Hwang, Yu-Ling Hsueh, and Yu-Ting Chen. An effective taxi recommender system based on a spatio-temporal factor analysis model. *Information Sciences*, 314:28–40, 2015.

[10] M Chris Jones. Simple boundary correction for kernel density estimation. *Statistics and computing*, 3:135–146, 1993.

[11] Diederik Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational diffusion models. *Advances in neural information processing systems*, 34:21696–21707, 2021.

[12] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[13] Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

[14] Diederik P Kingma, Max Welling, et al. An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning*, 12(4):307–392, 2019.

[15] Durk P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised learning with deep generative models. *Advances in neural information processing systems*, 27, 2014.

[16] Yifei Ma, Balakrishnan Narayanaswamy, Haibin Lin, and Hao Ding. Temporal-contextual recommendation in real-time. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2291–2299, 2020.

[17] Raghvendra Mall, Rocco Langone, and Johan AK Suykens. Self-tuned kernel spectral clustering for large scale networks. In *2013 IEEE International Conference on Big Data*, pages 385–393. IEEE, 2013.

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

[19] Northern California Earthquake Data Center. UC Berkeley Seismological Laboratory. Dataset. NCEDC, 2014.

[20] Yosihiko Ogata. On lewis' simulation method for point processes. *IEEE transactions on information theory*, 27(1):23–31, 1981.

[21] Yosihiko Ogata. Space-time point-process models for earthquake occurrences. *Annals of the Institute of Statistical Mathematics*, 50:379–402, 1998.

[22] Takahiro Omi, naonori ueda, and Kazuyuki Aihara. Fully neural network based model for general temporal point processes. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[23] Alex Reinhart. A review of self-exciting spatio-temporal point processes and their applications. *Statistical Science*, 33(3):299–318, 2018.

[24] Oleksandr Shchur, Marin Biloš, and Stephan Günnemann. Intensity-free learning of temporal point processes. In *International Conference on Learning Representations*, 2020.

[25] Oleksandr Shchur, Ali Caner Türkmen, Tim Januschowski, and Stephan Günnemann. Neural temporal point processes: A review. *arXiv preprint arXiv:2104.03528*, 2021.

[26] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 2256–2265, Lille, France, 07–09 Jul 2015. PMLR.

[27] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. *Advances in neural information processing systems*, 28, 2015.

[28] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[29] Robert Earl Wallace. The san andreas fault system, california: An overview of the history, geology, geomorphology, geophysics, and seismology of the most well known plate-tectonic boundary in the world. 1990.

[30] Alex Williams, Anthony Degleris, Yixin Wang, and Scott Linderman. Point process models for sequence detection in high-dimensional neural spike trains. *Advances in neural information processing systems*, 33:14350–14361, 2020.

[31] David Wilmot and Frank Keller. A temporal variational model for story generation. *arXiv preprint arXiv:2109.06807*, 2021.

[32] Yong Yu, Xiaosheng Si, Changhua Hu, and Jianxun Zhang. A review of recurrent neural networks: Lstm cells and network architectures. *Neural computation*, 31(7):1235–1270, 2019.

[33] Shixiang Zhu, Haoyun Wang, Xiuyuan Cheng, and Yao Xie. Neural spectral marked point processes. In *International Conference on Learning Representations*, 2022.

[34] Shixiang Zhu and Yao Xie. Spatiotemporal-textual point processes for crime linkage detection. *The Annals of Applied Statistics*, 16(2):1151–1170, 2022.

A Model estimation

To learn the model, one can maximize the log-likelihood of the observed event series. The log-likelihood of observing a sequence with N_T events can therefore be obtained by

$$\ell(x_1, \ldots, x_{N_T}) = \int_{\mathcal{X}} \log \lambda(x|\mathcal{H}_{t(x)}) d\mathbb{N}(x) - \int_{\mathcal{X}} \lambda(x|\mathcal{H}_{t(x)}) dx. \tag{A1}$$

An equivalent form of this objective can be expressed using conditional PDF, as shown in the following equation (see Appendix B for the derivation):

$$\max_{\theta \in \Theta} \ \ell(\theta) \coloneqq \frac{1}{K} \sum_{k=1}^{K} \int_{\mathcal{X}} \ \log f_\theta(x|\mathcal{H}_{t(x)}) \ d\mathbb{N}_k(x), \tag{A2}$$

where K represents the total number of observed event sequences and $\mathbb{N}_k$ is the counting measure of the k-th event sequence. It is worth noting that this learning objective circumvents the need to compute the integral in the second term of (A1), which can be computationally intractable when events exist in a multi-dimensional data space.

Now the key challenge is *how do we obtain the conditional PDF of an event x without access to the function f_θ*? This is a commonly posed inquiry in the realm of generative model learning, and there are several pre-existing learning algorithms intended for generative models that can provide solutions to this question [3]. In the rest of this section, we present two learning strategies that approximate the conditional PDF using generated samples and demonstrate the effectiveness of the proposed approach using numerical examples.

Non-parametric density estimation We present a non-parametric learning strategy that approximates the conditional PDF using kernel density estimation (KDE). Specifically, the conditional PDF of the i-th event x_i can be estimated by,

$$f_\theta(x_i|\mathcal{H}_{t_i}) \approx \frac{1}{L} \sum_{l=1}^{L} \kappa_\sigma(x_i - \widetilde{x}_i^{(l)}), \tag{A3}$$

where $\{\widetilde{x}_i^{(l)}\}_{l=1}^{L}$ is a set of samples generated by model $g(\cdot, \boldsymbol{h}_{i-1})$ and κ_σ is a kernel function with a bandwidth σ. See our implementation details in Appendix C.

We note that it is important to consider boundary correction [10] for the kernel function in the time dimension, as the support of the next event's time is $[0, +\infty)$, and a regular KDE would extend it to negative infinity. To select the kernel bandwidth σ, we adopt a common approach called the *self-tuned kernel* [4, 17]. This method dynamically determines a value of σ for each sample $\widetilde{x}^{(j)}$ by computing the k-nearest neighbor (kNN) distance among other generated samples. The use of self-tuned kernels is crucial for the success of the model because the event distribution may change significantly over the training iterations. Therefore, adapting the bandwidth for each iteration and sample is necessary to achieve an accurate estimate of the conditional PDF.

Variational approximation Variational method is another widely-adopted approach for learning a wide spectrum of generative models. Examples of such models include variational autoencoders [13, 14] and diffusion models [8, 11, 26]. In this paper, we follow the idea of conditional variational autoencoder (CVAE) [27] and approximate the log conditional PDF using its evidence lower bound (ELBO):

$$\log f_\theta(x_i|\mathcal{H}_{t_i}) \geq -D_{\mathrm{KL}}(q(z|x_i, \boldsymbol{h}_{i-1})||p_\theta(z|\boldsymbol{h}_{i-1})) + \mathbb{E}_{q(z|x_i, \boldsymbol{h}_{i-1})} \left[\log p_\theta(x_i|z, \boldsymbol{h}_{i-1})\right], \tag{A4}$$

where q is a variational approximation of the posterior distribution over the random noise given observed i-th event x_i and its history $\boldsymbol{h}_{i-1}$. The first term on the right-hand side is the Kullback–Leibler (KL) divergence of the approximate posterior $q(\cdot|x_i, \boldsymbol{h}_{i-1})$ from the exact posterior $p_\theta(\cdot|\boldsymbol{h}_{i-1})$). The second term is the log-likelihood of the latent data generating process. The complete derivation of (A4) and implementation details can be found in the Appendix D.

B Derivation of the conditional probability of point processes

The conditional probability of point processes can be derived from the conditional intensity (1). Suppose we are interested in the conditional probability of events at a given point $x \in \mathcal{X}$, and we assume that there are i events that happen before $t(x)$. Let $\Omega(x)$ be a small neighborhood containing x. According to (1), we can rewrite $\lambda(x|\mathcal{H}_{t(x)})$ as following:

$$\begin{aligned}\lambda(x|\mathcal{H}_{t(x)}) &= \mathbb{E}\left(d\mathbb{N}(x)|\mathcal{H}_{t(x)}\right)/dx = \mathbb{P}\{x_{i+1} \in \Omega(x)|\mathcal{H}_{t(x)}\}/dx \\ &= \mathbb{P}\{x_{i+1} \in \Omega(x)|\mathcal{H}_{t_{i+1}} \cup \{t_{i+1} \geq t(x)\}\}/dx \\ &= \frac{\mathbb{P}\{x_{i+1} \in \Omega(x), t_{i+1} \geq t(x)|\mathcal{H}_{t_{i+1}}\}/dx}{\mathbb{P}\{t_{i+1} \geq t(x)|\mathcal{H}_{t_{i+1}}\}}.\end{aligned}$$

Here $\mathcal{H}_{t_{i+1}} = \{x_1, \dots, x_i\}$ represents the history up to i-th events. If we let $F(t(x)|\mathcal{H}_{t(x)}) = \mathbb{P}(t_{i+1} < t(x)|\mathcal{H}_{t_{i+1}})$ be the conditional cumulative probability, and $f(x|\mathcal{H}_{t(x)}) \triangleq f(x_{i+1} \in \Omega(x)|\mathcal{H}_{t_{i+1}})$ be the conditional probability density of the next event happening in $\Omega(x)$. Then the conditional intensity can be equivalently expressed as

$$\lambda(x|\mathcal{H}_{t(x)}) = \frac{f(x|\mathcal{H}_{t(x)})}{1 - F(t(x)|\mathcal{H}_{t(x)})}.$$

We multiply the differential $dx = dtdm$ on both sides of the equation and integral over the mark space $\mathcal{M}$:

$$\begin{aligned}dt \cdot \int_{\mathcal{M}} \lambda(x|\mathcal{H}_{t(x)})dm = \frac{dt \cdot \int_{\mathcal{M}} f(x|\mathcal{H}_{t(x)})dm}{1 - F(t(x)|\mathcal{H}_{t(x)})} &= \frac{dF(t(x)|\mathcal{H}_{t(x)})}{1 - F(t(x)|\mathcal{H}_{t(x)})} \\ &= -d\log\left(1 - F(t(x)|\mathcal{H}_{t(x)})\right).\end{aligned}$$

Hence, integrating over t on $[t_i, t(x))$ leads to the fact that

$$\begin{aligned}F(t(x)|\mathcal{H}_{t(x)}) &= 1 - \exp\left(-\int_{t_i}^{t(x)} \int_{\mathcal{M}} \lambda(x|\mathcal{H}_{t(x)})dmdt\right) \\ &= 1 - \exp\left(-\int_{[t_i, t(x))\times\mathcal{M}} \lambda(x|\mathcal{H}_{t(x)})dx\right)\end{aligned}$$

because $F(t_i) = 0$. Then we have

$$f(x|\mathcal{H}_{t(x)}) = \lambda(x|\mathcal{H}_{t(x)}) \cdot \exp\left(-\int_{[t_i, t(x))\times\mathcal{M}} \lambda(x|\mathcal{H}_{t(x)})dx\right),$$

which corresponds to (2).

The log-likelihood of one observed event series in (A1) is derived, by the chain rule, as

$$\begin{aligned}\ell(x_1, \dots, x_{N_T}) = \log f(x_1, \dots, x_{N_T}) &= \log \prod_{i=1}^{N_T} f(x_i|\mathcal{H}_{t_i}) \\ &= \int_{\mathcal{X}} \log f(x|\mathcal{H}_{t(x)})d\mathbb{N}(x) \\ &= \int_{\mathcal{X}} \log \lambda(x|\mathcal{H}_{t(x)})d\mathbb{N}(x) - \int_{\mathcal{X}} \lambda(x|\mathcal{H}_{t(x)})dx.\end{aligned}$$

The log-likelihood of K observed event sequences in (A2) can be conveniently obtained with the counting measure $\mathbb{N}$ replaced by the counting measure $\mathbb{N}_k$ for the k-th sequence.

C Implementation details of non-parametric learning

Estimating the conditional PDF $f(x|\mathcal{H}_{t(x)})$ using kernel density estimation (KDE) within our framework presents two main challenges: (1) The distribution density of events generated by certain inhomogeneous point processes can vary from location to location in the event space. Consequently, using a single bandwidth for estimation would either oversmooth the conditional PDF or introduce excessive noise in areas with sparse events. (2) The time intervals of the next events are usually clustered in a small neighborhood of 0 and always positive, which will lead to a significant boundary bias.

To overcome the above challenges, we adopt the self-tuned kernel with boundary correction:

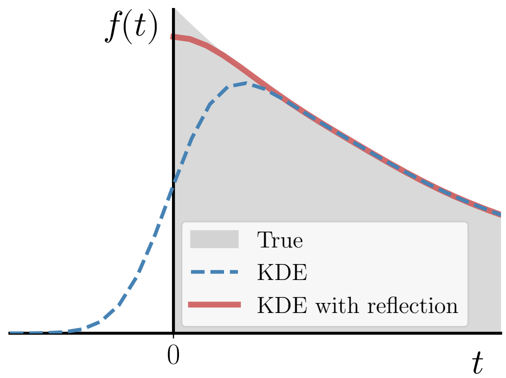
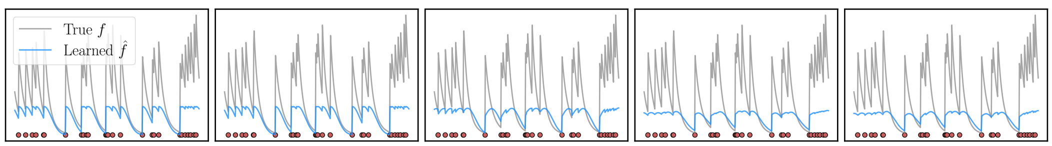
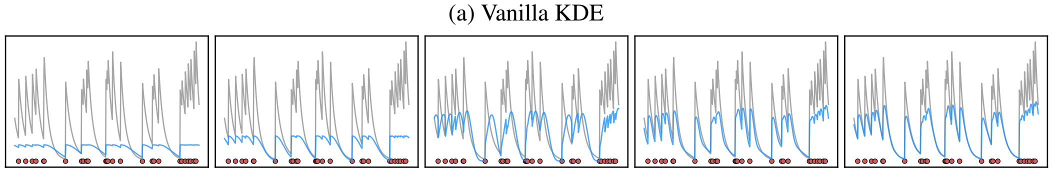


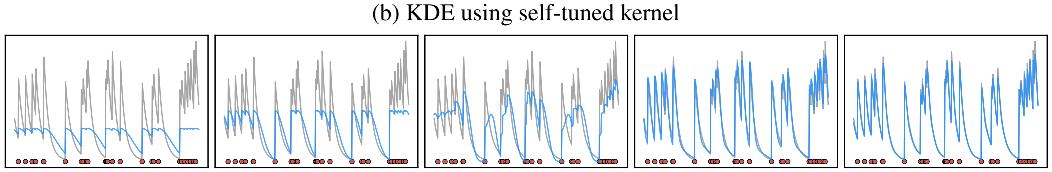
Figure B1: A comparison between the vanilla KDE and the KDE with boundary correction. The grey shaded area indicates the true density function, which is defined on the bounded region $[0, +\infty)$. The blue dashed line and red line show the estimated density function by the vanilla KDE and the KDE with reflection, respectively.



(a) Vanilla KDE



(b) KDE using self-tuned kernel



(c) KDE using self-tuned kernel with boundary correction

Figure B2: The estimated conditional PDF $f(t|\mathcal{H}_t)$ of a testing sequence is displayed from left to right. Each panel within the same row represents the estimated conditional PDF at intervals of 10 training epochs.

1. We first choose the bandwidth adaptively, where the bandwidth σ tends to be small for those samples falling into event clusters and to be large for those isolated samples. We dynamically determine the value of σ for each sample $\widetilde{x}$ by computing the k-nearest neighbor (kNN) distance among other generated samples [4, 17].
2. We correct the boundary bias of KDE by reflecting the data points against the boundary 0 in time domain [10]. Specifically, the kernel with reflection is defined as follows:

$$\kappa(x - \widetilde{x}) = \upsilon^*(\Delta t - \Delta\widetilde{t}) \cdot \upsilon(m - \widetilde{m}),$$

   where υ is an arbitrary kernel and $\upsilon^*(x - \widetilde{x}) = \upsilon(x - \widetilde{x}) + \upsilon(-x - \widetilde{x})$ is the same kernel with reflection boundary. This allows for a more accurate estimation of the density near the boundary of the time domain without impacting the estimation elsewhere, as shown in Figure B1.

Figure B2 compares the learned conditional PDF using three KDE methods on the same synthetic data set generated by a self-exciting Hawkes process. The results show that the estimation using the self-tuned kernel with boundary correction shown in (c) significantly outperforms two ablation models in (a) and (b). We also summarize the learning algorithm in Algorithm 2.

Algorithm 2 Non-parametric learning for CEG

Input: Training set X with K sequences: $X = \{x_i^{(k)}\}_{i=1,\ldots,\mathbb{N}_k(\mathcal{X}),\ k=1,\ldots,K}$, learning epoch E, learning rate γ, mini-batch size M.
Initialization: model parameters θ, $e = 0$
while $e < E$ **do**
  for each sampled batch $\widehat{X}^M$ with size M **do**
    1. Draw samples z from noise distribution $\mathcal{N}(0, 1)$;
    2. Feed z into the generator g to obtain sampled events $\widetilde{x}$;
    3. Estimate conditional PDF using KDE (A3) and log-likelihood ℓ (A1), given data $\widehat{X}^M$, samples $\widetilde{x}$ and the model;
    4. $\theta \leftarrow \theta + \gamma \partial \ell / \partial \theta$;
  end for
  $e \leftarrow e + 1$;
end while
return θ

D Derivation and implementation details of variational learning

Derivation of the approximate conditional PDF Now we present the derivation of the approximate conditional PDF in (A4). We first use hidden embedding $\boldsymbol{h}$ to represent the history $\mathcal{H}_t(x)$ and $f_\theta(x|\mathcal{H}_{t(x)})$ can be substituted by $f_\theta(x|\boldsymbol{h})$. Then the conditional PDF of event x given the history can be re-written as:

$$\log f_\theta(x|\boldsymbol{h}) = \log \int p_\theta(x, z|\boldsymbol{h}) dz,$$

where z is a latent random variable. This integral has no closed form and can usually be estimated by Monte Carlo integration with importance sampling, *i.e.*,

$$\int p_\theta(x, z|\boldsymbol{h}) dz = \mathbb{E}_{z \sim q(\cdot|x,\boldsymbol{h})} \left[\frac{p_\theta(x, z|\boldsymbol{h})}{q(z|x, \boldsymbol{h})} \right].$$

Here $q(z|x, \boldsymbol{h})$ is the proposed variational distribution, where we can draw sample z from this distribution given x and $\boldsymbol{h}$. Therefore, by Jensen's inequality, we can find the evidence lower bound (ELBO) of the conditional PDF:

$$\log f_\theta(x|\boldsymbol{h}) = \log \mathbb{E}_{z \sim q(\cdot|x,\boldsymbol{h})} \left[\frac{p_\theta(x, z|\boldsymbol{h})}{q(z|x, \boldsymbol{h})} \right] \geq \mathbb{E}_{z \sim q(\cdot|x,\boldsymbol{h})} \left[\log \frac{p_\theta(x, z|\boldsymbol{h})}{q(z|x, \boldsymbol{h})} \right].$$

Using Bayes rule, the ELBO can be equivalently expressed as:

$$\begin{aligned}
\mathbb{E}_{z \sim q(\cdot|x,\boldsymbol{h})} \left[\log \frac{p_\theta(x, z|\boldsymbol{h})}{q(z|x, \boldsymbol{h})} \right] &= \mathbb{E}_{z \sim q(\cdot|x,\boldsymbol{h})} \left[\log \frac{p_\theta(x|z, \boldsymbol{h}) p_\theta(z|\boldsymbol{h})}{q(z|x, \boldsymbol{h})} \right] \\
&= \mathbb{E}_{z \sim q(\cdot|x,\boldsymbol{h})} \left[\log \frac{p_\theta(z|\boldsymbol{h})}{q(z|x, \boldsymbol{h})} \right] + \mathbb{E}_{z \sim q(\cdot|x,\boldsymbol{h})} \left[\log p_\theta(x|z, \boldsymbol{h}) \right] \\
&= -D_{\text{KL}}(q(z|x, \boldsymbol{h}) || p_\theta(z|\boldsymbol{h})) + \mathbb{E}_{z \sim q(\cdot|x,\boldsymbol{h})} \left[\log p_\theta(x|z, \boldsymbol{h}) \right].
\end{aligned}$$

Implementation details In practice, we introduce two additional generator functions, *encoder net* $g_{\text{encode}}(\epsilon, x_i, \boldsymbol{h}_{i-1})$ and *prior net* $g_{\text{prior}}(\epsilon, \boldsymbol{h}_{i-1})$, respectively, to represent $q(z|x_i, \boldsymbol{h}_{i-1})$ and $p_\theta(z|\boldsymbol{h}_{i-1})$ as transformations of another random variable $\epsilon \sim \mathcal{N}(0, I)$ using reparametrization trick [26]. Both $q(z|x_i, \boldsymbol{h}_{i-1})$ and $p_\theta(z|\boldsymbol{h}_{i-1})$ are often modeled as Gaussian distributions, which allows us to compute the KL divergence of Gaussians with a closed-form expression. The log-likelihood of the second term can be implemented as the reconstruction loss and calculated using generated samples.

We parameterize both $p_\theta(z|\boldsymbol{h})$ and $q(z|x, \boldsymbol{h})$ using fully-connected neural networks with one hidden layer, denoted by g_{prior} and g_{encode}, respectively. The prior of the latent variable is modulated by the input $\boldsymbol{h}$ in our formulation; however, the constraint can be easily relaxed to make the latent variables statistically independent of input variables, *i.e.*, $p_\theta(z|\boldsymbol{h}) = p_\theta(z)$ [15, 27]. For the approximate posterior $q(z|x, \boldsymbol{h})$, a common choice is a simple factorized Gaussian encoder, which can be represented as:

$$q(z|x, \boldsymbol{h}) = \mathcal{N}(z; \mu, \text{diag}(\Sigma)),$$

Algorithm 3 Variational learning for CEG using stochastic gradient descent

Input: Training set X with K sequences: $X = \{x_i^{(k)}\}_{i=1,\ldots,\mathbb{N}_k(\mathcal{X}),\ k=1,\ldots,K}$, learning epoch E, learning rate γ, mini-batch size M.
Initialization: model parameters θ, $e = 0$
while $e < E$ **do**
  for each sampled batch $\widehat{X}^M$ with size M **do**
    1. Draw samples ϵ from noise distribution $\mathcal{N}(0, 1)$;
    2. Compute z using reparametrization trick, given data $\widehat{X}^M$, noise ϵ, g_{prior}, and g_{encode};
    3. Compute ELBO (A4) and log-likelihood ℓ (A1) based on z and data $\widehat{X}^M$;
    4. $\theta \leftarrow \theta + \gamma \partial\ell/\partial\theta$;
  end for
  $e \leftarrow e + 1$;
end while
return θ

or

$$q(z|x, \boldsymbol{h}) = \prod_{j=1}^{r} q(z_j|x, \boldsymbol{h}) = \prod_{j=1}^{r} \mathcal{N}(z_j; \mu_j, \sigma_j^2).$$

The Gaussian parameters $\mu = (\mu_j)_{j=1,\ldots,r}$ and $\text{diag}(\Sigma) = (\sigma_j^2)_{j=1,\ldots,r}$ are the output of an encoder network ϕ and the latent variable z can be obtained using reparametrization trick:

$$(\mu, \log \text{diag}(\Sigma)) = \phi(x, \boldsymbol{h}),$$
$$z = \mu + \text{diag}(\Sigma) \odot \epsilon,$$

where $\epsilon \sim \mathcal{N}(0, I)$ is another random variable and $\odot$ is the element-wise product. For simplicity in presentation, we denote such a factorized Gaussian encoder as $g_{\text{encode}}(\epsilon, x, \boldsymbol{h})$ that maps an event x, its history $\boldsymbol{h}$, and a random noise vector ϵ to a sample z from the approximate posterior for that event x.

In (A4), the first term is the KL divergence of the approximate posterior from the prior, which acts as a regularizer, while the second term is an expected negative reconstruction error. They can be calculated as follows: (1) Because both $q(z|x_i, \boldsymbol{h}_{i-1})$ and $p_\theta(z|\boldsymbol{h}_{i-1})$ are modeled as Gaussian distributions, the KL divergence can be computed using a closed-form expression. (2) Minimizing the negative log-likelihood $p_\theta(x|z, \boldsymbol{h})$ is equivalent to maximizing the cross entropy between the distribution of an observed event x and the reconstructed event $\widetilde{x}$ generated by the generative model g given z and the history $\boldsymbol{h}$. The learning algorithm has been summarized in Algorithm 3.

E Sampling efficiency comparison

Thinning algorithms are known to be challenging and suffer from low sampling efficiency. This is because (i) these algorithms require sampling uniformly in the space $\mathcal{X}$ with the upper limit of the conditional intensity $\overline{\lambda} > \lambda(x),\ \forall x$, and only a few candidate points are retained in the end. (ii) the decision of whether to reject one candidate point requires the evaluation of the conditional intensity function over the entire history, which is also stochastic. This doubly stochastic trait makes the entire thinning process particularly costly when $\mathcal{X}$ is a multi-dimensional space, since it requires a drastically large number of candidate points and numerous evaluations of the conditional intensity function.

On the contrary, our model generates samples based on the underlying conditional distribution of events learned from true data, thus every generated point will be retained. Table E1 compares the time costs for ETAS, DNSK, and CEG to generate event series of length 100 on each data set. Particularly noteworthy is that our model requires a similar amount of time to generate different numbers of sequences. This is because CEG can generate all the sequences in parallel, leveraging the benefits of the implementation of conditional generative models.

Table E1: Computation costs for generating earthquake series and time-stamped image series of length 100 using ETAS, DNSK and CEG.

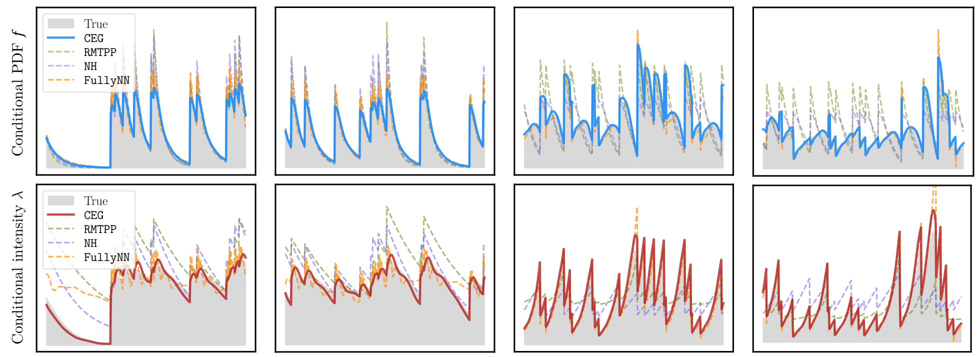
	3D earthquake data		T-MNIST		T-CIFAR	
Model	5 sequences	50 sequences	5 sequences	50 sequences	5 sequences	50 sequences
ETAS	12.4	118.6	/	/	/	/
DNSK	20.1	220.4	87.3	745.6	274.0	1381.9
CEG	< 1	< 1	0.6	0.8	1.1	1.2

*Unit: second.

F Experiment details and additional results

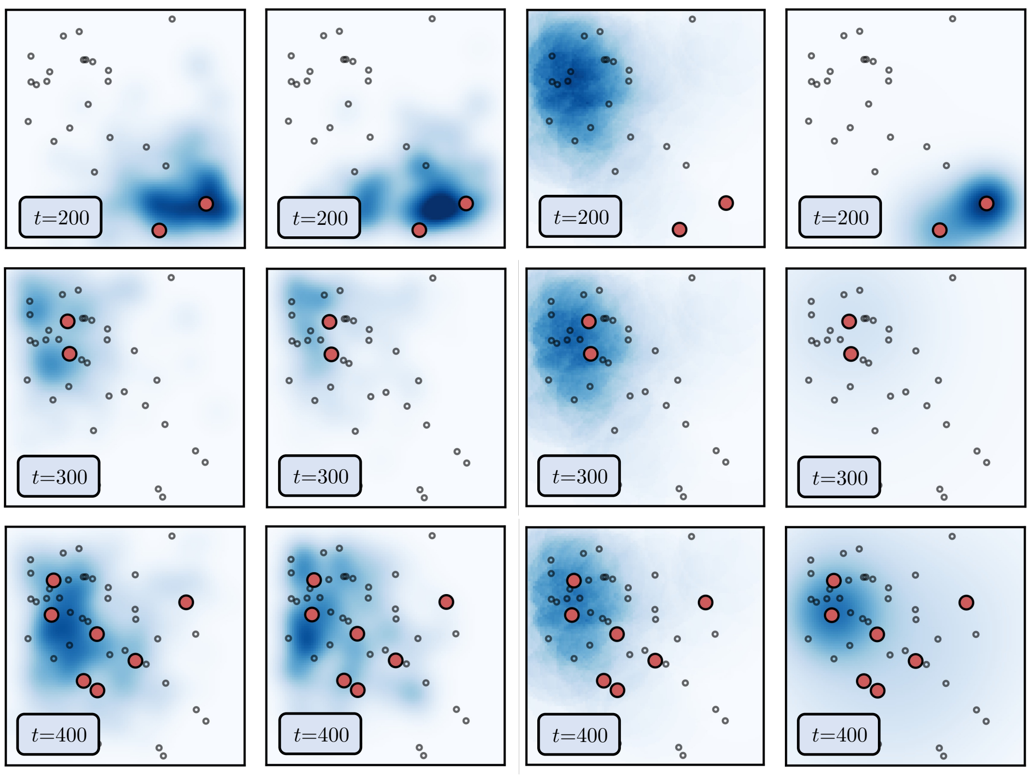
Baselines We compare our proposed method empirically with the following baselines:

1. *Recurrent Marked Temporal Point Process* (RMTPP) [6] uses an RNN to capture the nonlinear relationship between both the markers and the timings of past events. It models the conditional intensity function by
$$\lambda(t|\mathcal{H}_t) = \exp(\boldsymbol{v}^\top \boldsymbol{h}_i + w(t - t_i) + b),$$
where hidden state $\boldsymbol{h}_i$ of the RNN represents the event history until the nearest i-th event $\mathcal{H}_{t_i} \cup \{t_i\}$. The $\boldsymbol{v}, w, b$ are trainable parameters. The model is learned by MLE using backpropagation through time (BPTT).
2. *Neural Hawkes Process* (NH) [18] extends the classical Hawkes process by memorizing the long-term effects of historical events. The conditional intensity function is given by
$$\lambda(t|\mathcal{H}_t) = f(\boldsymbol{w}^\top \boldsymbol{h}_t),$$
where $\boldsymbol{h}_t$ is a sufficient statistic of the event history modeled by the hidden state in a continuous-time LSTM, and $f(\cdot)$ is a scaled softplus function for ensuring positive output. The weight $\mathbf{w}$ is learned jointly with the LSTM through MLE.
3. *Fully Neural Network based Model* (FullyNN) for General Temporal Point Processes [22] models the cumulative hazard function given the history embedding $\boldsymbol{h}_i$, which leads to a tractable likelihood. It uses a fully-connect neural network Z_i with a non-negative activation function for the cumulative hazard function $\Phi(\tau|\boldsymbol{h}_i)$ where $\tau = t - t_i$. The conditional intensity function is obtained by computing the derivative of the network:
$$\lambda(t|\mathcal{H}_t) = \frac{\partial}{\partial(\tau)}\Phi(\tau|\boldsymbol{h}_i) = \frac{\partial}{\partial(\tau)} Z_i(\tau),$$
where Z_i is the fully-connect neural network.
4. *Epidemic-type aftershock sequence* (ETAS) acts as a benchmark in spatio-temporal point process modeling. Denoting each event $x := (t, s)$, ETAS adopts a Gaussian diffusion kernel in the conditional intensity as following
$$\lambda(t, s|\mathcal{H}_t) = \mu + \sum_{(t_i, s_i)\in\mathcal{H}_t} k(t, t_i, s, s_i),$$
where
$$k(t, t_i, s, s_i) = \frac{Ce^{-\beta(t-t_i)}}{2\pi\sqrt{|\Sigma|}(t - t_i)} \cdot \exp\left\{-\frac{(s - s_i - a)^\top \Sigma^{-1}(s - s_i - a)}{2(t - t_i)}\right\}.$$
Here $\Sigma = \mathrm{diag}(\sigma_x^2, \sigma_y^2)$ is a diagonal matrix representing the covariance of the spatial correlation. Note that the diffusion kernel is stationary and only depends on the spatio-temporal distance between two events. All the parameters are learnable.
5. *Deep non-stationary kernel* (DNSK) proposes a neural-network-based influence kernel based on kernel singular value decomposition for modeling spatio-temporal point process data. In addition, their kernel can be extended to handle high-dimensional marks:
$$k(t_i, t-t_i, s_i, s-s_i, m_i, m) = \sum_{q=1}^{Q}\sum_{r=1}^{R}\sum_{l=1}^{L} \alpha_{lrq}\psi_l(t_i)\varphi_l(t-t_i)u_r(s_i)v_r(s-s_i)g_q(m_i)h_q(m).$$
Here all the basis functions are represented by fully-connected neural networks.



(a) Self-exciting 1 (b) Self-exciting 2 (c) Self-correcting 1 (d) Self-correcting 2

Figure F3: Out-of-sample estimation of the conditional PDF $f(t|\mathcal{H}_t)$ and the corresponding intensity $\lambda(t|\mathcal{H}_t)$ using the proposed method on one-dimensional (time only) synthetic event sequences.



(a) Ground truth (b) Proposed `CEG` (c) `DNSK` (d) `ETAS`

Figure F4: Snapshots of out-of-sample estimation of the conditional PDFs for a three-dimensional (time and space) synthetic event sequence, arranged in chronological order from left to right. The conditional PDFs are indicated by shaded areas, with darker shades indicating higher conditional PDF values. The red dots represent newly observed events within the most recent time period, while the circles represent historical events.

Synthetic data description We use the following point process models to generate the one-dimensional synthetic data sets using Algorithm 4:

1. Self-exciting Hawkes process: $\lambda(t) = \mu + \sum_{t_i \in \mathcal{H}_t} \beta e^{-\beta(t-t_i)}$, with $\mu = 0.1, \beta = 0.1$ and $\mu = 0.5, \beta = 1.0$ in self-exciting data 1 and 2, respectively.
2. Self-correcting process: $\lambda(t) = \exp\left(\mu t - \sum_{t_i \in \mathcal{H}_t} \alpha\right)$, with $\mu = 1.0, \alpha = 1.0$ and $\mu = 0.5, \alpha = 0.8$ in self-correcting data 1 and 2, respectively.

Algorithm 4 Thinning algorithm

Input: Model $\lambda(\cdot)$, time horizon T, mark space $\mathcal{M}$, Intensity upper bound $\bar{\lambda}$.
Initialization: $\mathcal{H}_T = \emptyset, t = 0, i = 0$
while $t < T$ **do**
  1. Sample $u \sim \text{Unif}(0, 1)$.
  2. $t \leftarrow t - \ln u/\bar{\lambda}$.
  3. Sample $m \sim \text{Unif}(\mathcal{M}), D \sim \text{Unif}(0, 1)$.
  4. $\lambda = \lambda(t, m|\mathcal{H}_T)$.
  if $D\bar{\lambda} \leq \lambda$ **then**
    $i \leftarrow i + 1; t_i = t, m_i = m$.
    $\mathcal{H}_T \leftarrow \mathcal{H}_T \cup \{(t_i, m_i)\}$.
  end if
end while
if $t_i \geq T$ **then**
  return $\mathcal{H}_T - \{(t_i, m_i)\}$
else
  return $\mathcal{H}_T$
end if

3. T-MNIST: In the MNIST series, all the digits that are greater than nine will be truncated to nine. The exponentially decaying kernel for the observation times are $k(t, t_i) = \beta e^{-\beta(t-t_i)}, \beta = 0.2$.
4. T-CIFAR: The images of bicycles and motorcycles represent outdoor exercises; the apples, pears, and oranges represent food ingestion; the computer keyboards represent study/working; and the beds represent sleeping. Before 21:00, the activity series progresses with the transition probability matrix between (exercise, food ingestion, working) being

$$P = \begin{pmatrix} 0.0 & 1.0 & 0.0 \\ 0.2 & 0.0 & 0.8 \\ 0.2 & 0.3 & 0.5 \end{pmatrix}.$$

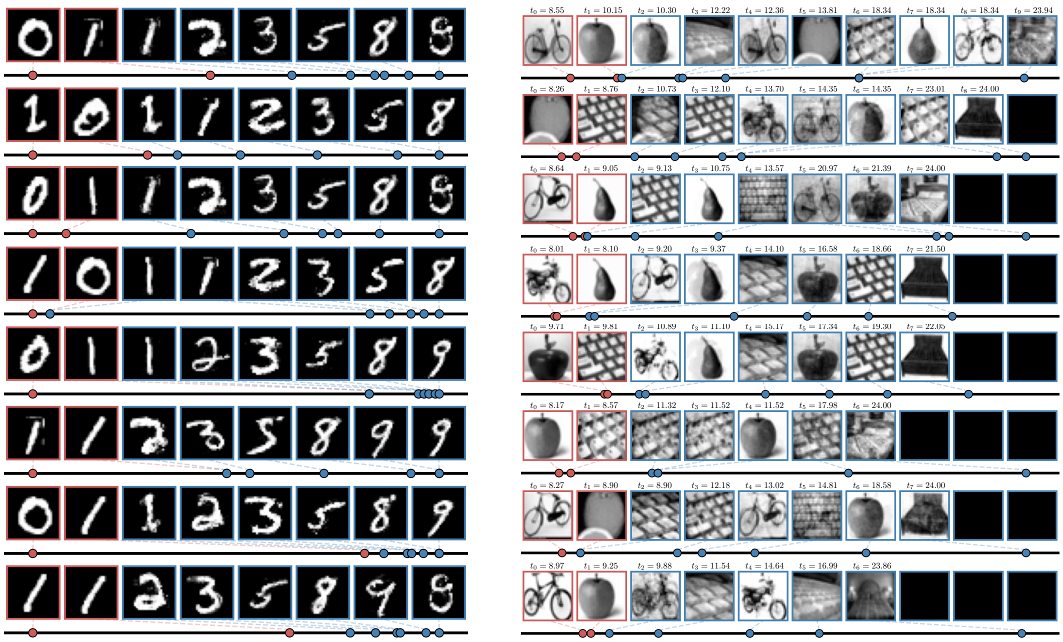
Starting from 21:00, the probability of sleeping increases linearly from 0 to 1 at 23:00. Each series ends with the activity of sleeping. The self-correcting process for event times is set with $\mu = 0.1, \alpha = 0.5$, indicating that each activity will last for a while before the student moves to the next activity (or stays in the current one).

Experimental setup We choose our generator g to be a fully-connected neural network with two hidden layers of width 32 with softplus activation function. To guarantee that the generated time interval is always positive, we apply an extra Rectified Linear Unit (ReLU) function for the output of the time dimension in the output layer. We use an LSTM for the history encoder ψ. We train our model and other baselines using 90% of the data and test them on the remaining 10% data. To fit the model parameters, we maximize log-likelihood according to (A2), and adopt Adam optimizer [12] with a learning rate of 10^{-3} and a batch size of 32 (event sequences). More details about experimental setup can be found in Appendix F.

For `RMTPP`, `NH` and `FullyNN`, we take the default parameters for model architectures in the original papers, with the dimension of hidden embedding to be 64 for all three models, and a fully-connected neural network with two hidden layers of width 64 for the cumulative hazard function in `FullyNN`. There is no hyperparameter in `ETAS`. All the baselines are trained using the Adam optimizer with a learning rate of 10^{-3} and a batch size of 32 for 100 epochs. The experiments are implemented on Google Colaboratory (Pro version) with 12GB RAM and a Tesla T4 GPU.

F.1 Additional experiment results

3D synthetic data Each row in Figure F4 displays four snapshots of estimated conditional probability density functions (PDFs) for a particular 3D testing sequence. It is apparent that our model's estimated PDFs closely match the ground truth and accurately capture the complex spatial and temporal point patterns. Conversely, `DNSK` and `ETAS` model for estimating spatio-temporal point processes fails to capture the heterogeneous triggering effects between events, indicating limited practical representational power.

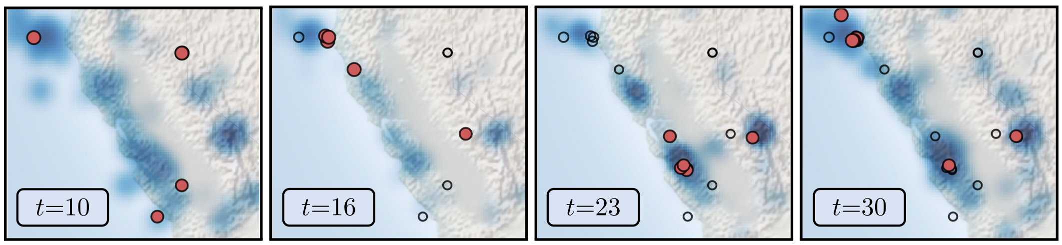


(a) Additional T-MNIST series generated by `CEG` (b) Additional T-CIFAR series generated by `CEG`

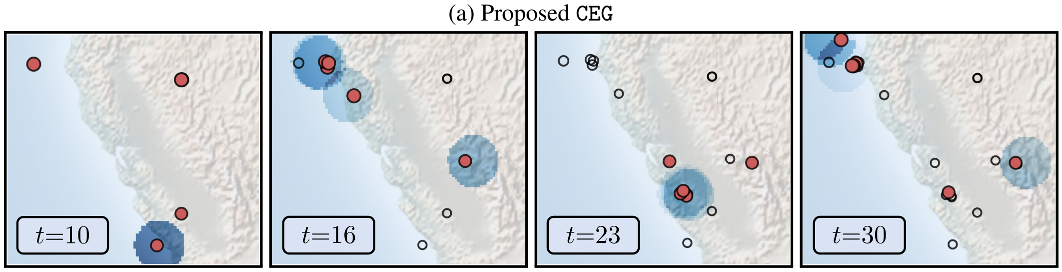
Figure F5: Additional T-MNIST and T-CIFAR series using `CEG` and a neural point process baseline `DNSK`, with true sequences displayed on the left. Each event series is generated (blue boxes) given the first two true events (red boxes).

Semi-synthetic image data More generated T-MNIST and T-CIFAR series by `CEG` are presented in Figure F5. As we can see, our generative point process can not only sample images that resemble the ground truth, but also recover the intricate temporal dynamics (*e.g.*, clustering effect of self-exciting process in T-MNIST, student's sleeping time in T-CIFAR) and high-dimensional mark dependencies.

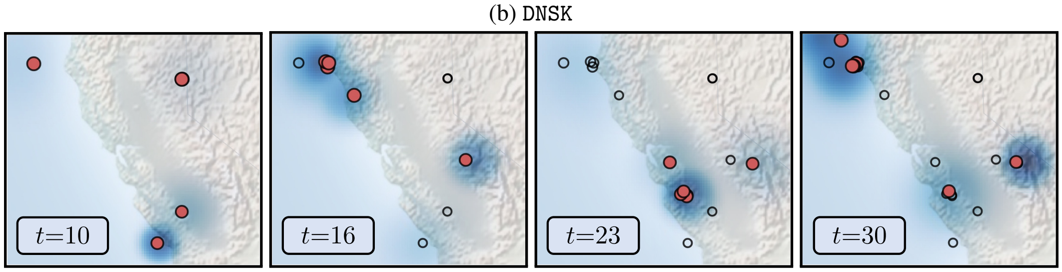
Northern California earthquake catalog Additional results in Figure F6 visualizes the conditional PDF estimated by `CEG`, `DNSK`, and `ETAS` for an actual earthquake sequence in testing set, respectively. The results indicate that our model is able to capture the heterogeneous effects among earthquakes. Particularly noteworthy is our model's finding of a heightened probability of seismic activity along the San Andreas fault, coupled with a diminished likelihood in the basin. These results align with current understandings of the mechanics of earthquakes in Northern California. However, both `DNSK` and `ETAS` fail to extract this geographical feature from the data and suggest that observed earthquakes impact their surroundings uniformly, leading to an increased likelihood of aftershocks within a circular area centered on the location of the initial event.



(a) Proposed CEG



(b) DNSK



(c) ETAS

Figure F6: Estimated conditional PDFs of an actual earthquake sequence represented by shaded areas, with darker shades indicating higher conditional PDF values. Each row contains four sub-figures, arranged in chronological order from left to right, showing snapshots of the estimated conditional PDFs. The red dots represent newly observed events within the most recent time period, while the circles represent historical events.