Arbitrarily Conditioned Hierarchical Flows for Spatiotemporal Events

Keyan Chen
Kahlert School of Computing
University of Utah
u1466725@utah.edu

Qiwei Yuan
Kahlert School of Computing
University of Utah
joshua.yuan@utah.edu

Zhitong Xu
Kahlert School of Computing
University of Utah
u1502956@utah.edu

Bin Shen
Celonis AI
stanshenbin@gmail.com

Shandian Zhe*
Kahlert School of Computing
University of Utah
zhe@cs.utah.edu

Abstract

Events in spatiotemporal systems are ubiquitous, yet modeling their complex distributions remains challenging. Existing point process models often rely on strong structural assumptions and are typically limited to autoregressive, event-by-event prediction. As a result, they struggle to support broader inference tasks such as inverse inference, trajectory reconstruction, and recovery of missing event locations. We introduce Arbitrarily Conditioned Hierarchical Flows (ARCH), a hierarchical flow matching framework for spatiotemporal event modeling. ARCH is expressive enough to capture complex event distributions while enabling tractable and accurate computation of conditional intensities, which quantify instantaneous event risk. Built on a history-encoder-generative-decoder architecture, ARCH introduces a hybrid masking strategy for flexible conditioning on arbitrary observed events. This enables a unified treatment of forecasting, inverse inference, and partial trajectory recovery within a single framework. Experiments on synthetic and real-world datasets demonstrate the consistent advantages of ARCH across prediction and conditional inference tasks. Our code is publicly available at https://github.com/BayesianAIGroup/ARCH.

1 Introduction

Spatiotemporal events arise in many domains, including weather phenomena, natural disasters, epidemics, and crime. Their distributions are often highly complex, while observations are typically sparse, making learning and inference over such events particularly challenging.

Existing spatiotemporal point process models often rely on strong structural assumptions, such as the independent-increment property of Poisson processes, additive interaction structures in Hawkes-process-like models (Yuan et al., 2019; Zhou et al., 2022; Zhou and Yu, 2023), and parametric distribution families (Shchur et al., 2020). More recent approaches have explored diffusion-based event generation (Yuan et al., 2023), which substantially improves expressive power for capturing complex distributions. However, this flexibility often comes at the cost of losing tractable or well-calibrated conditional intensities, a central quantity for quantifying instantaneous event risk. Furthermore, existing methods are typically limited to autoregressive, event-by-event inference, making them ill-suited for broader tasks such as inverse inference, event trajectory reconstruction, and recovery of missing event locations.

*Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

To overcome these bottlenecks, we propose ARCH, a hierarchical flow matching framework for spatiotemporal event modeling. Our method is flexible enough to capture complex event distributions from data while also enabling efficient and accurate computation of conditional intensities. In addition, it supports arbitrarily conditioned inference, allowing a broad range of tasks, including forecasting, inverse inference, missing-event imputation, and partial trajectory recovery. The major contributions of our work are summarized as follows:

- **Model.** We propose a hierarchical flow matching framework that first generates the event time and then, conditioned on the event time, generates the event spatial location. This decomposed generation procedure not only retains the flexibility needed to capture complex distributions, but also enables tractable and accurate computation of the conditional intensity, which quantifies instantaneous event risk. Specifically, we leverage the Picard–Lindelöf theorem and the order-preserving property of one-dimensional ODEs to compute the conditional intensity of the event-time flow, and then solve the companion log-probability ODE for the event-location flow, combining the two to obtain the final conditional event intensity.
- **Hybrid Masked Training.** To support a variety of inference tasks, we employ a history encoder and a generative decoder for each flow. We further introduce a hybrid masking strategy that enables parallel autoregressive generation, randomly conditioned inference, batched trajectory recovery, and future trajectory forecasting. During training, masks are continuously sampled, and the model is trained to recover the target velocity field for masked events conditioned on partially observed events. In this way, a broad range of inference tasks is covered, with training and inference unified within a single framework.
- **Experiments.** On three real-world benchmark datasets, ARCH demonstrates strong performance across diverse tasks, including next-event prediction, initial-event inference, missing-event imputation, partial-trajectory recovery, and probability density estimation. It also enables joint multi-step forecasting, achieving comparable or superior accuracy to its autoregressive counterpart and diffusion-based autoregressive baselines. On synthetic datasets generated from diverse spatiotemporal point processes, ARCH achieves more accurate conditional intensity recovery, with lower relative L_2 errors than competing structured point process models. Extensive ablation studies further validate the key design choices of our framework.

2 Preliminaries

Flow matching (Lipman et al., 2022) learns a continuous-time vector field that transports a simple source distribution to a target data distribution through an ordinary differential equation (ODE). Let $\mathbf{x}_0 \sim p_0$ denote a random sample from a tractable source distribution, typically a standard Gaussian, and let $\mathbf{x}_1 \sim p_{\mathrm{data}}$ denote a data sample. The goal is to construct a time-dependent probability path $\{p_t\}_{t\in[0,1]}$ that smoothly connects p_0 and p_{data}, and to learn a velocity field $\mathbf{v}_\theta(\mathbf{x}, t)$ such that samples evolve according to the ODE:

$$\frac{d\mathbf{x}(t)}{dt} = \mathbf{v}_\theta(\mathbf{x}(t), t), \qquad t \in [0, 1]. \tag{1}$$

If the learned velocity field matches the true velocity field associated with the probability path, then integrating the ODE from an initial noise sample $\mathbf{x}(0) = \mathbf{x}_0$ transports $\mathbf{x}_0$ to a sample $\mathbf{x}(1)$ distributed according to p_{data}. A common choice in flow matching is to define a conditional probability path between paired samples $(\mathbf{x}_0, \mathbf{x}_1)$. One simple and widely used example is the linear flow, which interpolates between source and target as $\mathbf{x}_t = (1-t)\mathbf{x}_0 + t\mathbf{x}_1$ where $t \in [0, 1]$. Under this construction, the corresponding conditional velocity is constant along the path: $\mathbf{u}_t(\mathbf{x}_1 \mid \mathbf{x}_0) = \frac{d\mathbf{x}_t}{dt} = \mathbf{x}_1 - \mathbf{x}_0$. Flow matching trains $\mathbf{v}_\theta$ to match this target velocity field along sampled intermediate points. In practice, one samples a data point $\mathbf{x}_1 \sim p_{\mathrm{data}}$, a noise sample $\mathbf{x}_0 \sim p_0$, and a time $t \sim \mathrm{Uniform}(0, 1)$, constructs the interpolated point $\mathbf{x}_t$, and minimizes the regression loss

$$\mathcal{L}_{\mathrm{FM}}(\theta) = \mathbb{E}_{t,\mathbf{x}_0,\mathbf{x}_1}\left[\left\|\mathbf{v}_\theta(\mathbf{x}_t, t) - (\mathbf{x}_1 - \mathbf{x}_0)\right\|^2\right]. \tag{2}$$

More generally, other probability paths can be used, with corresponding target velocities derived from the chosen transport construction.

After training, generation is performed by first drawing the initial state $\mathbf{x}(0) \sim p_0$ from the source distribution, and then solving the learned ODE (1) forward in time. The final state $\mathbf{x}(1)$ is taken

as a generated sample. In practice, the ODE is solved numerically using a standard solver, such as Euler or higher-order Runge-Kutta methods. Thus, once the velocity field has been learned, sampling reduces to deterministic transport from noise to data through the learned continuous-time dynamics.

3 Methodology

Consider a sequence of spatiotemporal events $\Gamma = [(s_1, \mathbf{x}_1), \ldots, (s_N, \mathbf{x}_N)]$, where $s_1 < \cdots < s_N$, each $s_n \in \mathbb{R}_+$ denotes the timestamp of event n, and $\mathbf{x}_n \in \mathbb{R}^d$ denotes its spatial location. Given a collection of observed event sequences $\mathcal{D}$, our goal is to learn the underlying distribution governing the generation of such sequences, while supporting a broad range of inference tasks, including forecasting future events, inverse inference of initial events, imputing missing event times or locations, and reconstructing partial event trajectories.

3.1 Hierarchical Flow-Matching Framework

We propose a hierarchical flow-matching framework that flexibly captures complex event distributions while also enabling tractable and accurate computation of conditional intensities, which quantify instantaneous event risk and play a central role in event modeling tasks such as risk monitoring, survival analysis, hotspot discovery, and intervention planning Cox (1972); Daley and Vere-Jones (2003); Braga (2001). The overall architecture is illustrated in Appendix Figure 2.

To illustrate the framework, consider generating a future event $\mathbf{e} = (s, \mathbf{x})$ given the history sequence $\mathcal{H} = [(s_1, \mathbf{x}_1), \ldots, (s_M, \mathbf{x}_M)]$. Instead of constructing a single flow that simultaneously generates the event time $s \in \mathbb{R}_+$ and spatial location $\mathbf{x} = (x_1, \ldots, x_d) \in \mathbb{R}^d$, we introduce two flows, $\mathcal{G}_1$ and $\mathcal{G}_2$: $\mathcal{G}_1$ generates s conditioned on $\mathcal{H}$, while $\mathcal{G}_2$ generates $\mathbf{x}$ conditioned on the generated time s and $\mathcal{H}$. This yields the decomposition

$$p(s, \mathbf{x} \mid \mathcal{H}) = p(s \mid \mathcal{H})\, p(\mathbf{x} \mid s, \mathcal{H}), \tag{3}$$

where $\mathcal{G}_1$ models $p(s \mid \mathcal{H})$ and $\mathcal{G}_2$ models $p(\mathbf{x} \mid s, \mathcal{H})$. Note that $\mathcal{G}_1$ and $\mathcal{G}_2$ are not restricted to autoregressive next-event generation; as explained in Section 3.2, they support arbitrarily conditioned generation more generally.

A key advantage of this design is that it enables efficient and accurate computation of the conditional intensity, i.e., the instantaneous event risk[2]:

$$\lambda(s, \mathbf{x} \mid \mathcal{H}) = \frac{p(s, \mathbf{x} \mid \mathcal{H})}{1 - F_S(s \mid \mathcal{H})}, \tag{4}$$

where $F_S(\cdot \mid \mathcal{H})$ denotes the marginal cumulative distribution function (CDF) of the event time:

$$F_S(s \mid \mathcal{H}) = p(S \le s \mid \mathcal{H}) = \int_{\Omega \cap (s_M, s] \times (-\infty, \infty)^d} p(\tau, \mathbf{x} \mid \mathcal{H})\, d\tau\, \mathrm{d}x_1 \cdots \mathrm{d}x_d, \tag{5}$$

and Ω is the spatiotemporal domain of the events.

If one instead uses a single flow to generate s and $\mathbf{x}$ simultaneously, then computing the marginal CDF $F_S(s \mid \mathcal{H})$ in (4) from the joint density $p(s, \mathbf{x} \mid \mathcal{H})$ requires integrating over a $(d+1)$-dimensional spatiotemporal domain, as shown in (5). This integral typically does not admit a closed form, while numerical integration or Monte Carlo approximation can be computationally expensive and may introduce substantial approximation error. In contrast, our hierarchical flows induce the density decomposition in (3), which in turn yields:

$$\lambda(s, \mathbf{x} \mid \mathcal{H}) = \lambda(s \mid \mathcal{H})\, p(\mathbf{x} \mid s, \mathcal{H}), \qquad \lambda(s \mid \mathcal{H}) = \frac{p(s \mid \mathcal{H})}{1 - F_S(s \mid \mathcal{H})}. \tag{6}$$

Thus, the full conditional intensity factors into the marginal temporal intensity $\lambda(s \mid \mathcal{H})$ and the conditional spatial density $p(\mathbf{x} \mid s, \mathcal{H})$, which can be computed separately from $\mathcal{G}_1$ and $\mathcal{G}_2$.

We first consider $p(\mathbf{x} \mid s, \mathcal{H})$. Let the ODE associated with $\mathcal{G}_2$ be

$$\frac{d\mathbf{x}(t)}{dt} = \mathbf{v}_{\Theta_2}\big(\mathbf{x}(t), t \mid \mathcal{H}, s\big), \tag{7}$$

[2]More precisely, the conditional intensity, also called the hazard, characterizes the instantaneous likelihood of an event occurring at time s, conditioned on the event history and on no new event having occurred prior to s.

where $\mathbf{v}_{\Theta_2}(\cdot)$ denotes the vector field and Θ_2 its parameters. By the instantaneous change-of-variable formula, the log-density along the trajectory $\mathbf{x}(t)$ evolves according to

$$\frac{d}{dt}\log p(\mathbf{x}(t) \mid s, \mathcal{H}) = -\nabla \cdot \mathbf{v}_{\Theta_2}\big(\mathbf{x}(t), t \mid \mathcal{H}, s\big), \tag{8}$$

where $\nabla \cdot \mathbf{v}$ denotes the divergence of the velocity field. To compute the conditional density at a given value $\mathbf{x}_{\text{val}}$, we augment the flow (7) with (8) and define a joint state $\mathbf{z}(t) = [\mathbf{x}(t), \ell(t)]$, where $\ell(t)$ accumulates the change in log-density. We then set $\mathbf{x}(1) = \mathbf{x}_{\text{val}}$ and $\ell(1) = 0$, and solve the coupled ODEs backward from $t = 1$ to $t = 0$. This yields the corresponding noise sample $\mathbf{x}(0)$, which follows the prior distribution p_{prior}, together with the accumulated change $\ell(0) = \int_1^0 \frac{d}{dt}\log p(\mathbf{x}(t) \mid s, \mathcal{H})\, dt$. The final log-density is therefore $\log p(\mathbf{x} \mid s, \mathcal{H}) = \log p_{\text{prior}}\big(\mathbf{x}(0)\big) - \ell(0)$.

Next, we consider the marginal temporal intensity $\lambda(s \mid \mathcal{H})$. Given a specific value s_{val}, we follow a similar procedure to obtain the corresponding noise value $s(0)$ and the log-density $\log p(s_{\text{val}} \mid \mathcal{H})$. The remaining quantity is the marginal CDF $F_S(s_{\text{val}} \mid \mathcal{H})$, which can be obtained from the CDF of the noise variable at $s(0)$ according to the following lemma.

Lemma 3.1. *Let $S(0)$ be a real-valued random variable, and let $S(t)$ be the solution at time $t \geq 0$ to the one-dimensional ordinary differential equation $\frac{ds}{dt} = v(s(t), t)$, with random initial condition $S(0)$. Let $F_t(\cdot)$ denote the cumulative distribution function of $S(t)$. Then, for any deterministic solution trajectory $\{s(t)\}_{t\geq 0}$ of the ODE with initial value $s(0)$, the CDF evaluated along the trajectory is invariant in time:*

$$F_t(s(t)) = F_0(s(0)), \qquad \forall t \geq 0.$$

This can be established using the Picard–Lindelöf theorem together with the order-preserving property of one-dimensional ODEs; we defer the proof to Appendix A. Since the noise distribution is chosen to be simple, *e.g.,* a standard Gaussian, its CDF can be evaluated efficiently. In practice, to ensure accurate ODE solutions, we can use higher-order Runge-Kutta methods, such as DOPRI5 (Dormand and Prince, 1980).

3.2 Arbitrarily Conditioned Training and Inference

To support a broad range of inference tasks beyond standard autoregressive, event-by-event prediction, we introduce a hybrid masking strategy for arbitrarily conditioned training and inference. Specifically, each flow employs a history encoder and a generative decoder, both implemented as Transformers. During training, given an event sequence $\Gamma = [(s_1, \mathbf{x}_1), \ldots, (s_N, \mathbf{x}_N)]$, the encoder treats each event $e_n = (s_n, \mathbf{x}_n) \in \Gamma$ as an input token and produces event representations through self-attention layers. The encoder serves as the conditioning branch, providing integrated feature representations of the observed events. These representations are then incorporated into the decoder through cross-attention to guide generation of the target events. The decoder also contains N tokens, each corresponding to one generated event. The input to decoder token n consists of the flow time t and the intermediate ODE state at time t. For the flow $\mathcal{G}_2$ that models spatial locations, *i.e.,* $p(\mathbf{x} \mid s, \mathcal{H})$, the decoder input at token n additionally includes the time s_n of event n.

To flexibly support diverse inference tasks, we combine several masking strategies during training:

Autoregressive Mask. This mask supports standard autoregressive generation. We impose a causal self-attention mask on the encoder so that each event token i attends only to itself and past events $j < i$. On the decoder side, token n attends to encoder tokens $1, \ldots, n-1$. This cross-attention pattern allows decoder token n to generate event n conditioned on its entire preceding history. Combined with causal self-attention in the encoder, this design enables parallel autoregressive training over all events in the sequence.

Random Condition Mask. This mask supports generation of an arbitrary subset of events conditioned on the remaining observed events. Specifically, we sample a binary condition mask $\mathbf{c} = (c_1, \ldots, c_N)$, where each $c_n \in \{0, 1\}$, $c_n = 1$ indicates that event n is conditioned (observed), and $c_n = 0$ indicates that event n is to be generated. On the encoder side, we mask out all tokens with $c_n = 0$ and perform full self-attention among the conditioned events to exchange and fuse their representations. On the decoder side, we mask out all tokens with $c_n = 1$ and perform self-attention among the events to be generated. We then set the cross-attention mask $\mathcal{M}[n, m] = 0$ for all pairs (n, m) such that $c_n = 0$

and $c_m = 1$, allowing every generated token to attend to every conditioned token. In this way, the full conditioned subset facilitates joint generation of the target events.

Consecutive Mask. This mask supports generation of a consecutive subsequence of events conditioned on the remaining events, which is useful for trajectory reconstruction and joint multi-step forecasting. Although the random condition mask can in principle cover such cases, its independently sampled entries more often produce scattered missing events rather than contiguous missing segments. To address this, we randomly sample two endpoints a, b with $1 \leq a < b \leq N$, set $c_n = 0$ for $a \leq n \leq b$, and set $c_n = 1$ otherwise. This yields a mask in which a consecutive block of events is generated conditioned on the events outside the block.

During training, we sample a mini-batch of event sequences and apply each masking strategy to compute the corresponding flow-matching loss. The temporal flow is trained to generate log-transformed inter-event times rather than absolute timestamps, which removes dependence on the arbitrary time origin, compresses large temporal gaps, and improves numerical stability. The objective is to recover the target velocity field on the tokens corresponding to events that need to be generated. For the random-conditioning and consecutive masks, the mask pattern is sampled independently for each sequence in the batch. We then sum the resulting losses across all mask types and backpropagate through the full objective to update the model parameters. The training/generation procedures are summarized in Appendix Algorithms 1–2. The masking strategies are illustrated in Figure 3.

4 Related Work

Temporal Event Modeling. A large body of work has developed temporal point process models for temporal event data. Popular approaches include Poisson processes and Hawkes processes (HPs) (Hawkes, 1971; Blundell et al., 2012; Lloyd et al., 2015; Zhe and Du, 2018; Pan et al., 2021; Wang et al., 2020; Pan et al., 2020; Wang et al., 2022; Xu et al., 2026b). Later efforts moved toward learning conditional intensity functions with deep neural networks. Representative examples include Neural Hawkes Processes (NHP) (Mei and Eisner, 2017), which encode event histories using LSTMs (Hochreiter and Schmidhuber, 1997), Recurrent Marked Temporal Point Processes (RMTPP) (Du et al., 2016), which similarly use recurrent architectures while explicitly modeling event marks, and Transformer-based approaches (Zhang et al., 2020; Zuo et al., 2020). In addition, Mei et al. (2019) addressed missing-event imputation in point processes via bidirectional particle smoothing, while Gupta et al. (2021) proposed jointly modeling observed and latent events to infer missing events.

More recent research has explored modern generative models for temporal event generation. Lin et al. (2022) proposed an autoregressive diffusion approach that generates events one at a time. Lüdke et al. (2023) introduced a diffusion framework that uses a homogeneous Poisson process as the noise distribution and applies thinning and deletion operations to edit sequences during generation. Lüdke et al. (2026) further developed edit-based generation by introducing several sequence edit operations to align noise sequences with event sequences of potentially different lengths, and used a flow model to learn the corresponding edit-rate table. Kerrigan et al. (2026) first generates the sequence length and then generates the event sequence accordingly. Despite their expressive power, these methods generally do not provide tractable or well-calibrated conditional intensities, making it difficult to recover the instantaneous event risk that is central to standard event modeling.

Spatiotemporal Event Modeling. Classical spatiotemporal point process models (Yuan et al., 2019; Reinhart, 2018) largely extend Poisson and Hawkes processes to the spatiotemporal setting. More recently, Chen et al. (2021) proposed a neural spatiotemporal point process that models jumps in the conditional intensity upon event arrivals using recurrent neural networks, and employs neural ordinary differential equations (ODEs) (Chen et al., 2018) to capture continuous-time intensity evolution between events. In contrast, Jia and Benson (2019) used neural stochastic differential equations (SDEs) to model latent state dynamics governing the conditional intensity. Other recent works retain the additive, excitation-only structure of spatiotemporal Hawkes processes while increasing modeling flexibility. The model of Zhou et al. (2022) parameterizes the triggering kernel using Transformer encodings of historical events. Zhou and Yu (2023) introduced a neural triggering kernel based on monotonic networks (Sill, 1997), which allows exact likelihood integration. Xu et al. (2026a) modeled event influences using Gaussian processes to capture both excitation and inhibition, and leveraged Kronecker-product algebra and tensor-product quadrature for efficient, tractable training

and inference. Yuan et al. (2023) proposed an autoregressive diffusion model that sequentially generates spatiotemporal events. Although this method demonstrates strong performance on next-event prediction, it, like the temporal generative models above, does not explicitly provide calibrated conditional intensities. Overall, existing spatiotemporal event models either rely on restrictive structural assumptions or prioritize expressive generation without retaining tractable conditional intensities, and are typically focused on autoregressive next-event prediction rather than more general conditioned inference tasks.

5 Experiments

5.1 Intensity Recovery

We first evaluated whether ARCH can recover the conditional intensity of event sequences, i.e., the instantaneous event risk. To this end, we constructed two synthetic spatiotemporal point processes. The first is a spatiotemporal Hawkes process with conditional intensity

$$\lambda(s, \mathbf{x}|\mathcal{H}_s) = \lambda_0 + \sum\nolimits_{s_n<s} \alpha e^{-\beta(s-s_n)} \frac{1}{2\pi\sigma^2} e^{-\frac{\|\mathbf{x}-\mathbf{x}_n\|^2}{2\sigma^2}}, \tag{9}$$

where $s \in [0, 60]$ denotes time, $\mathbf{x} \in [0, 1]^2$ denotes the spatial location, and $\mathcal{H}_s = \{(s_n, \mathbf{x}_n) \mid s_n < s\}$ denotes the history of events before time s. We set $\lambda_0 = 1.0$, $\alpha = 0.72$, $\beta = 1.2$, and $\sigma = 0.05$.

The second process is a spatiotemporal self-correcting process,

$$\lambda(s, \mathbf{x}|\mathcal{H}_s) = \mu \exp\left(\alpha s \phi(\mathbf{x}|c_0) - \sum\nolimits_{s_n<s} \beta\phi(\mathbf{x}_n - \mathbf{x}|c_1)\right), \tag{10}$$

where $\phi(\cdot|c) = \mathcal{N}(\cdot|\mathbf{0}, c\mathbf{I})$. We set $\mu = 1.0$, $\alpha = 0.2$, $\beta = 0.4$, $c_0 = 0.25$, and $c_1 = 0.2$. The spatiotemporal domain was set to $(s, \mathbf{x}) \in [0, 100] \times [0, 1]^2$. We refer to the first process as **STHP-SYN** and the second as **STSC-SYN**. For each process, we simulated 10K event sequences for training, 1K for validation, and 200 for testing using Ogata's thinning algorithm (Ogata, 1981).

Table 1: Relative L_2 error of the conditional intensity on synthetic datasets. Best results are shown in bold.

Model	**STHP-SYN**	**STSC-SYN**
AutoSTPP	0.572 ± 0.187	0.588 ± 0.153
DeepSTPP	0.631 ± 0.126	0.918 ± 0.037
NSTPP	0.981 ± 0.022	0.819 ± 0.087
ARCH	$\mathbf{0.458 \pm 0.171}$	$\mathbf{0.370 \pm 0.132}$

We compared ARCH with several state-of-the-art spatiotemporal point process models that explicitly parameterize the conditional intensity function. The first baseline is Deep Spatiotemporal Hawkes Process (DeepSTPP) (Zhou et al., 2022), which adopts a Hawkes-style intensity with exponential triggering kernels, where the kernel parameters are predicted by a Transformer encoder over past events. The second is Automatic Integration for Spatiotemporal Neural Point Processes (AutoSTPP) (Zhou and Yu, 2023), which extends Hawkes-style intensities with neural triggering kernels parameterized by monotonic networks (Sill, 1997), enabling exact likelihood integration. The third is Neural Spatiotemporal Point Process (NSTPP) (Chen et al., 2021), which models the conditional intensity using neural ODEs and updates the event-history representation through GRU-based state transitions when new events arrive. Our method was implemented in PyTorch and trained using the AdamW optimizer (Loshchilov and Hutter, 2017) with a learning rate of 10^{-3} for up to 300 epochs. The mini-batch size was set to 64. The two flows in our model share the same Transformer backbone but use separate heads for velocity prediction. For conditional intensity evaluation, we solved the associated ODEs using DOPRI5 (Dormand and Prince, 1980) with a fixed integration budget of 20 solver steps. Additional training details and hyperparameter settings are provided in the Appendix Section C. For DeepSTPP, AutoSTPP, and NSTPP, we used the official implementations and default hyperparameter settings released by the original authors.

For each test sequence, we evaluated the conditional intensity at every event time. At each time point, we computed an intensity snapshot on a uniform 8×8 spatial grid. We then computed the relative L_2 error of the predicted snapshots against the ground-truth snapshots, aggregated over all evaluated time points. As shown in Table 1, ARCH achieves substantially smaller relative L_2 errors than the competing approaches, demonstrating more accurate recovery of the conditional intensity.

We then randomly selected one test sequence from **STHP-SYN** and **STSC-SYN**, and visualized the learned intensity functions at five representative time points. As shown in Figure 1 and Appendix

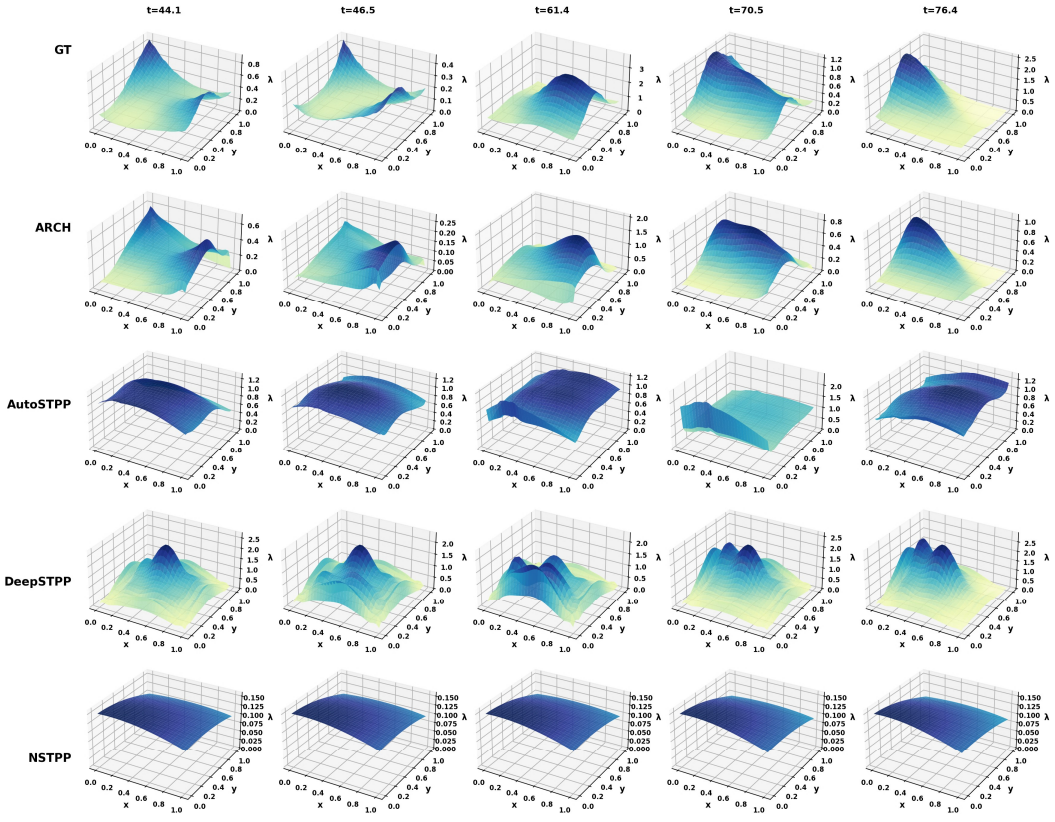


Figure 1: Conditional intensity functions at representative time points on **STSC-SYN**. GT: ground-truth.

Figure 4, ARCH closely recovers the shape of the ground-truth intensity across all time points. DeepSTPP and AutoSTPP also capture the overall intensity shape on **STHP-SYN** reasonably well; see Appendix Figure 4. However, their learned intensities deviate substantially from the ground truth on **STSC-SYN**. This is likely because both methods impose an additive, excitation-only Hawkes-style intensity structure, whereas **STSC-SYN** is generated from a multiplicative intensity model with strong inhibitory interactions among events. The mismatch can therefore be attributed, at least in part, to model misspecification. Moreover, even on **STHP-SYN**, where their recovered intensity shapes are visually close to the ground-truth, DeepSTPP and AutoSTPP exhibit more pronounced scale discrepancies than ARCH, which is consistent with their larger relative L_2 errors in Table 1. In contrast, ARCH faithfully captures the intensity structure on both datasets, demonstrating its flexibility in capturing both excitation-dominated and inhibition-driven spatiotemporal dynamics without relying on a restrictive Hawkes-style parameterization. Finally, NSTPP consistently exhibits poor recovery performance: its learned intensity surfaces show only minor variation across time points. This suggests that NSTPP struggles to learn a time-sensitive conditional spatial distribution in these synthetic settings. Additional visualizations of the marginal temporal intensities (Appendix G, Figures 5 and 6) further corroborate the advantage of ARCH in intensity recovery.

5.2 Predictive Performance

We next evaluated ARCH on a range of prediction tasks using three real-world benchmark datasets: **Earthquake** (Chen et al., 2021), **COVID-19** (Chen et al., 2021), and **Citibike** (Yuan et al., 2023). These datasets capture seismic activity, infectious disease propagation, and urban mobility, providing diverse spatiotemporal event scenarios. Details of the datasets are provided in Appendix B.

Next Event Prediction. We first evaluated ARCH on next-event prediction, the standard forecasting task in spatiotemporal event modeling. We compared against several classical and recent state-of-the-art methods: (1) DeepSTPP (Zhou et al., 2022); (2) NSTPP (Chen et al., 2021); (3) Diffusion

Table 2: Root-mean-square error (RMSE) and Euclidean distance for next-event time and location prediction. The best two results are highlighted in bold.

Model	Earthquake		COVID-19		Citibike	
	Spatial ↓	Temporal ↓	Spatial ↓	Temporal ↓	Spatial ↓	Temporal ↓
HPP	9.45±0.000	0.412±0.000	0.818±0.000	0.113±0.000	0.452±0.000	0.239±0.000
STHP	8.35±0.252	0.424±0.018	0.422±0.000	0.100±0.001	0.032±0.000	0.633±0.126
NJSDE	9.98±0.024	0.465±0.009	0.641±0.009	0.137±0.001	0.707±0.001	0.264±0.005
NSTPP	8.11±0.000	0.547±0.010	0.560±0.000	0.145±0.002	0.705±0.000	0.355±0.013
DeepSTPP	9.20±0.000	**0.341±0.000**	0.687±0.000	0.197±0.000	0.044±0.000	0.234±0.000
DSTPP	**6.77±0.193**	**0.375±0.001**	**0.419±0.001**	**0.093±0.000**	**0.031±0.000**	**0.200±0.002**
ARCH (Ours)	**6.55±0.022**	**0.375±0.001**	**0.391±0.000**	**0.097±0.000**	**0.031±0.000**	**0.211±0.000**

Spatiotemporal Point Process (DSTPP) (Yuan et al., 2023), a recent diffusion-based generative model that autoregressively generates the next spatiotemporal event conditioned on the history. Due to its DDPM formulation (Ho et al., 2020), DSTPP cannot evaluate the conditional intensity; (4) Neural Jump Stochastic Differential Equations (NJSDE) (Jia and Benson, 2019), which encode event histories as latent states and model their dynamics with neural SDEs; (5) Spatiotemporal Hawkes Process (STHP), using the conditional intensity form in Eq. (9); and (6) Homogeneous Poisson Process (HPP). We implemented STHP in PyTorch and trained it with the AdamW optimizer using a learning rate of 10^{-3} for up to 1K epochs to ensure convergence. HPP admits a closed-form solution and does not require explicit training. The architecture of ARCH is kept fixed across all prediction tasks. For next-event prediction, ARCH uses the autoregressive mask and performs Euler integration with 10 steps. For all competing neural baselines, we used the public implementations released by the original authors.

We used the same training, validation, and test splits as provided in (Yuan et al., 2023) for all three datasets. Following the experimental protocol of (Chen et al., 2021; Yuan et al., 2023), we repeated each experiment five times with different random initializations. We evaluated temporal prediction using the root-mean-square error (RMSE) of the predicted event time, and spatial prediction using the Euclidean distance between the predicted and ground-truth event locations. We report the mean and standard deviation in Table 2. ARCH consistently achieves top performance in both temporal and spatial prediction. Its prediction accuracy is often close to that of the diffusion-based model DSTPP, while substantially outperforming other neural point process baselines in many cases. In particular, ARCH achieves the best spatial prediction accuracy and the second-best temporal prediction accuracy across all three datasets. Although ARCH and DSTPP obtain comparable next-event prediction performance, DSTPP cannot provide calibrated estimates of event risk through the conditional intensity and is restricted to autoregressive next-event generation.

Table 3: Inverse inference of the initial events.

Model	Earthquake		COVID-19		Citibike	
	Spatial ↓	Temporal ↓	Spatial ↓	Temporal ↓	Spatial ↓	Temporal ↓
KNNImp ($k = 3$)	12.2±0.000	0.506±0.000	0.972±0.000	0.174±0.000	0.052±0.000	0.462±0.000
KNNImp ($k = 5$)	11.3±0.000	0.518±0.000	0.830±0.000	0.172±0.000	0.046±0.000	0.456±0.000
KNNImp ($k = 10$)	10.2±0.000	0.513±0.000	0.723±0.000	0.173±0.000	0.042±0.000	0.456±0.000
NHPS	N/A	0.553±0.004	N/A	0.151±0.004	N/A	0.403±0.003
ARCH	**7.60±0.022**	**0.471±0.006**	**0.464±0.001**	**0.148±0.002**	**0.033±0.000**	**0.373±0.011**

Inverse Inference. Second, we evaluated ARCH on inverse inference, where the goal is to infer the initial (first two) events from subsequent observations. Existing spatiotemporal event models are not designed to support such inverse inference tasks, we therefore compared against KNNImpute (Troyanskaya et al., 2001), a general-purpose and widely used missing-value imputation method based on nearest neighbors. We used the implementation from Scikit-Learn (Pedregosa et al., 2011) and varied the number of neighbors $k \in \{3, 5, 10\}$. In addition, we compared against NHPS (Mei et al., 2019), a temporal-only, fixed-slot adaptation of Neural Hawkes Particle Smoothing specifically designed for missing event imputation. It uses bidirectional continuous-time LSTM proposals and eight importance-weighted particles under a Neural Hawkes target process; spatial results are thus N/A. As shown in Table 3, ARCH consistently outperforms KNNImpute and NHPS, demonstrating substantially better recovery of the initial events. These results highlight a key advantage of our arbitrarily conditioned formulation: the same trained model can be directly applied to nonstandard

inference tasks beyond autoregressive forecasting, without task-specific retraining or architectural modification.

Missing Event Recovery. Third, we evaluated ARCH on recovering randomly missing events. For each test sequence, we randomly selected a subset of events and inferred them conditioned on the remaining observations, with missing ratios of 10%, 20%, and 30%. As shown in Appendix Table 17, ARCH outperforms KNNImpute and NHPS in nearly all settings, with only a few exceptions in COVID-19 temporal prediction: KNNImpute achieves slightly lower error, as does NHPS at the 10% missing ratio. These results demonstrate the strong event imputation capabilities of ARCH, despite being designed as a general-purpose generative model rather than a specialized imputation method.

We further evaluate recovery of partially observed events, where either the event time or location is missing. ARCH outperforms NHPS in all evaluated cases and KNNImpute in nearly all cases, with only one tie and one minor shortfall (Appendix F, Table 18).

Table 4: Internal event trajectory recovery.

Model	Earthquake		COVID-19		Citibike	
	Spatial ↓	Temporal ↓	Spatial ↓	Temporal ↓	Spatial ↓	Temporal ↓
Recovery length = 5						
KNNImp ($k = 3$)	8.71±0.123	0.558±0.011	0.540±0.007	0.186±0.006	0.037±0.000	0.193±0.006
KNNImp ($k = 5$)	8.42±0.125	0.523±0.008	0.518±0.006	0.179±0.005	0.035±0.000	0.182±0.005
KNNImp ($k = 10$)	8.24±0.104	0.496±0.006	0.495±0.005	0.174±0.006	0.034±0.000	**0.174±0.005**
NHPS	N/A	0.536±0.012	N/A	0.188±0.006	N/A	0.183±0.005
ARCH	**7.90±0.127**	**0.456±0.010**	**0.467±0.007**	**0.173±0.008**	**0.031±0.000**	**0.176±0.005**
Recovery length = 10						
KNNImp($k = 3$)	8.91±0.060	0.548±0.008	0.536±0.006	0.214±0.005	0.037±0.000	0.196±0.002
KNNImp($k = 5$)	8.53±0.058	0.518±0.006	0.510±0.006	0.204±0.004	0.035±0.000	0.182±0.003
KNNImp($k = 10$)	8.25±0.045	0.494±0.006	0.490±0.006	0.198±0.004	0.034±0.000	0.169±0.003
NHPS	N/A	0.551±0.011	N/A	0.206±0.003	N/A	0.177±0.003
ARCH	**7.75±0.060**	**0.458±0.008**	**0.462±0.006**	**0.183±0.006**	**0.031±0.000**	**0.156±0.004**

Table 5: Forecasting future event trajectories.

Model	Earthquake		COVID-19		Citibike	
	Spatial ↓	Temporal ↓	Spatial ↓	Temporal ↓	Spatial ↓	Temporal ↓
Trajectory length = 5						
DSTPP	10.8±0.067	0.573±0.013	0.612±0.006	0.070±0.002	0.045±0.000	1.054±0.007
ARCH-AR	8.74±0.039	0.420±0.005	**0.446±0.004**	**0.064±0.003**	**0.032±0.000**	0.935±0.005
ARCH	**8.73±0.042**	**0.416±0.002**	0.448±0.004	0.071±0.004	**0.032±0.000**	**0.910±0.008**
Trajectory length = 10						
DSTPP	11.1±0.085	0.589±0.014	0.626±0.005	0.083±0.002	0.045±0.000	0.844±0.006
ARCH-AR	8.76±0.054	**0.463±0.002**	**0.459±0.004**	**0.068±0.002**	**0.032±0.000**	0.695±0.004
ARCH	**8.74±0.054**	0.465±0.006	0.461±0.003	0.087±0.005	**0.032±0.000**	**0.626±0.008**

Event Trajectory Recovery. Fourth, we evaluated ARCH on recovering a missing event trajectory, i.e., a contiguous subsequence within an event sequence. For each test sequence, we randomly masked out a subsequence and applied ARCH to predict the masked events conditioned on the remaining observations. We considered masked subsequences of length 5 and 10. As shown in Table 4, ARCH consistently outperforms KNNImpute and NHPS in all the cases, except that when the masked trajectory length is 5, where the temporal prediction error of ARCH is slightly higher than that of KNNImpute with $k = 10$. Notably, when the masked trajectory length increases to 10, ARCH consistently outperforms KNNImpute across all datasets and metrics. This suggests that ARCH is particularly effective at capturing longer-range dependencies and reconstructing coherent event trajectories from partial observations.

Forecasting Future Trajectories. Finally, we evaluated ARCH on jointly forecasting a trajectory of future events. We ran DSTPP for comparison. Since DSTPP is restricted to auto-regressive prediction, each future event is generated sequentially conditioned on the observed history and previously generated events. We also evaluated an autoregressive variant of our method, denoted by ARCH-AR, which forecasts future events one at a time using the same autoregressive procedure. Note that joint forecasting generates all k future events with one temporal and one spatial ODE solve, whereas

ARCH-AR repeats this two-stage procedure for each event. As shown in Table 5, both ARCH-AR and ARCH consistently outperform DSTPP, often by a large margin. The only exception is temporal prediction on COVID-19, where the joint forecasting variant of ARCH is slightly worse than DSTPP. However, when using the same autoregressive forecasting strategy, ARCH-AR still outperforms DSTPP. Together, these results demonstrate the stronger long-horizon forecasting capability of our method. Overall, ARCH achieves better or comparable accuracy relative to ARCH-AR, showing that joint forecasting can generate multiple future events more efficiently while retaining accuracy comparable to the standard autoregressive forecasting strategy used by existing methods.

5.3 Additional Results

Held-out likelihood. We further evaluated predictive density using test negative log-likelihood (NLL) in Appendix D. ARCH achieves the best NLL on COVID-19 and Citibike and the second-best on Earthquake, highlighting the strong density estimation capabilities of our hierarchical flows for spatiotemporal events.

Ablation studies. We conducted ablation studies to assess the key design choices of ARCH, including temporal–spatial hierarchy, flow matching, training masks, and encoder–decoder architecture (Appendices E.1, E.2, and E.3). The hierarchical generation framework maintains competitive predictive accuracy while enabling event risk evaluation without spatial marginalization. Flow matching achieves comparable or superior conditional prediction accuracy to direct regression, while additionally supporting predictive sampling and uncertainty quantification. Random-conditioning and consecutive masks improve scattered and contiguous event recovery, respectively. Finally, while a block-wise masked decoder-only architecture is a viable alternative, our encoder–decoder design requires less peak GPU memory.

6 Conclusion

We presented ARCH, a hierarchical flow-matching framework for spatiotemporal event modeling. ARCH combines the expressiveness of continuous generative modeling with tractable and accurate evaluation of conditional intensities and instantaneous event risks. Its flexible conditioning supports diverse inference and prediction tasks, including inverse inference, missing-event and missing-attribute completion, event-trajectory recovery, and future-trajectory forecasting. Future work will explore broader real-world applications and scale the framework toward general-purpose foundation models for spatiotemporal event data.

Acknowledgments

SZ acknowledges support from NSF CAREER Award IIS-2046295 (CAREER: Embedding High-Order Interaction Events: Models, Algorithms, and Applications) and NSF CSSI-2311685 (Elements: A Convergent Physics-based and Data-driven Computing Platform for Building Modeling), and NSF DMS-2529112 (Collaborative Research: MATH-DT: Computationally efficient hypercomplex variable-based sensitivity methods for rapid Digital Twin model updating).

References

Aalaila, Y., Vollmer, S., and Großmann, G. (2026). Seahorse: A unified benchmarking framework for spatiotemporal event modeling. In *Advances in Neural Information Processing Systems (NeurIPS 2026), Track on Evaluations and Datasets*. Preprint: arXiv:2607.01022.

Blundell, C., Beck, J., and Heller, K. A. (2012). Modelling reciprocating relationships with hawkes processes. In *Advances in Neural Information Processing Systems*, pages 2600–2608.

Braga, A. A. (2001). The effects of hot spots policing on crime. *The ANNALS of the American Academy of Political and Social Science*, 578(1):104–125.

Chen, R. T., Rubanova, Y., Bettencourt, J., and Duvenaud, D. K. (2018). Neural ordinary differential equations. *Advances in neural information processing systems*, 31.

Chen, R. T. Q., Amos, B., and Nickel, M. (2021). Neural spatio-temporal point processes. In *International Conference on Learning Representations*.

Coddington, E. A. and Levinson, N. (1955). *Theory of Ordinary Differential Equations*. McGraw-Hill.

Cox, D. R. (1972). Regression models and life-tables. *Journal of the royal statistical society: Series B (methodological)*, 34(2):187–202.

Daley, D. J. and Vere-Jones, D. (2003). An introduction to the theory of point processes, volume 1: Elementary theory and methods. *Verlag New York Berlin Heidelberg: Springer*.

Dormand, J. R. and Prince, P. J. (1980). A family of embedded Runge–Kutta formulae. *Journal of Computational and Applied Mathematics*, 6(1):19–26.

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*, pages 1555–1564.

Gupta, V., Bedathur, S., Bhattacharya, S., and De, A. (2021). Learning temporal point processes with intermittent observations. In *Proceedings of the 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pages 3790–3798. PMLR.

Hawkes, A. G. (1971). Spectra of some self-exciting and mutually exciting point processes. *Biometrika*, 58(1):83–90.

Ho, J., Jain, A., and Abbeel, P. (2020). Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851.

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

Jia, J. and Benson, A. R. (2019). Neural jump stochastic differential equations. *Advances in Neural Information Processing Systems*, 32.

Kerrigan, G., Nelson, K., and Smyth, P. (2026). EventFlow: Forecasting temporal point processes with flow matching. In *The 29th International Conference on Artificial Intelligence and Statistics*.

Lin, H., Wu, L., Zhao, G., Pai, L., and Li, S. Z. (2022). Exploring generative neural temporal point process. *Transactions on Machine Learning Research*.

Lipman, Y., Chen, R. T., Ben-Hamu, H., Nickel, M., and Le, M. (2022). Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*.

Lloyd, C., Gunter, T., Osborne, M., and Roberts, S. (2015). Variational inference for gaussian process modulated poisson processes. In *International Conference on Machine Learning*, pages 1814–1822. PMLR.

Loshchilov, I. and Hutter, F. (2017). Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Lüdke, D., Bilovs, M., Shchur, O., Lienen, M., and Günnemann, S. (2023). Add and thin: Diffusion for temporal point processes. *Advances in Neural Information Processing Systems*, 36:56784–56801.

Lüdke, D., Lienen, M., Kollovieh, M., and Günnemann, S. (2026). Edit-based flow matching for temporal point processes. In *The Fourteenth International Conference on Learning Representations*.

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

Mei, H., Qin, G., and Eisner, J. (2019). Imputing missing events in continuous-time event streams. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 4475–4485. PMLR.

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

Pan, Z., Wang, Z., Phillips, J. M., and Zhe, S. (2021). Self-adaptable point processes with nonparametric time decays. *Advances in Neural Information Processing Systems*, 34:4594–4606.

Pan, Z., Wang, Z., and Zhe, S. (2020). Scalable nonparametric factorization for high-order interaction events. In *International Conference on Artificial Intelligence and Statistics*, pages 4325–4335. PMLR.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, É. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

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

Shchur, O., Bilovs, M., and Günnemann, S. (2020). Intensity-free learning of temporal point processes. *International Conference on Learning Representations (ICLR)*.

Sill, J. (1997). Monotonic networks. *Advances in neural information processing systems*, 10.

Troyanskaya, O., Cantor, M., Sherlock, G., Brown, P., Hastie, T., Tibshirani, R., Botstein, D., and Altman, R. B. (2001). Missing value estimation methods for DNA microarrays. *Bioinformatics*, 17(6):520–525.

Wang, Z., Chu, X., and Zhe, S. (2020). Self-modulating nonparametric event-tensor factorization. In *International Conference on Machine Learning*, pages 9857–9867. PMLR.

Wang, Z., Xu, Y., Tillinghast, C., Li, S., Narayan, A., and Zhe, S. (2022). Nonparametric embeddings of sparse high-order interaction events. In *International Conference on Machine Learning*, pages 23237–23253. PMLR.

Xu, Z., Yuan, Q., Chen, Y., Sun, Y., Shen, B., and Zhe, S. (2026a). Kronecker-structured nonparametric spatiotemporal point processes. In Perković, E. and Malinsky, D., editors, *Proceedings of the 42nd Conference on Uncertainty in Artificial Intelligence*, volume 337 of *Proceedings of Machine Learning Research*, pages 7594–7612. PMLR.

Xu, Z., Yuan, Q., Chen, Y., Zhe, S., and Shen, B. (2026b). Structured neural marked point processes for interpretable event interaction modeling. *arXiv preprint arXiv:2605.17568*.

Yuan, B., Li, H., Bertozzi, A. L., Brantingham, P. J., and Porter, M. A. (2019). Multivariate spatiotemporal Hawkes processes and network reconstruction. *SIAM Journal on Mathematics of Data Science*, 1(2):356–382.

Yuan, Y., Ding, J., Shao, C., Jin, D., and Li, Y. (2023). Spatio-temporal diffusion point processes. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '23, page 3173–3184, New York, NY, USA. Association for Computing Machinery.

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

Zhe, S. and Du, Y. (2018). Stochastic nonparametric event-tensor decomposition. In *Advances in Neural Information Processing Systems*, pages 6856–6866.

Zhou, Z., Yang, X., Rossi, R., Zhao, H., and Yu, R. (2022). Neural point process for learning spatiotemporal event dynamics. In Firoozi, R., Mehr, N., Yel, E., Antonova, R., Bohg, J., Schwager, M., and Kochenderfer, M., editors, *Proceedings of The 4th Annual Learning for Dynamics and Control Conference*, volume 168 of *Proceedings of Machine Learning Research*, pages 777–789. PMLR.

Zhou, Z. and Yu, R. (2023). Automatic integration for spatiotemporal neural point processes. In Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., and Levine, S., editors, *Advances in Neural Information Processing Systems*, volume 36, pages 50237–50253. Curran Associates, Inc.

Zuo, S., Jiang, H., Li, Z., Zhao, T., and Zha, H. (2020). Transformer hawkes process. In *International Conference on Machine Learning*, pages 11692–11702. PMLR.

Appendix

A Proof of Lemma 3.1

Proof. Let $\Phi_t : \mathbb{R} \to \mathbb{R}$ denote the flow map induced by the ODE, so that for each initial value $s_0 \in \mathbb{R}$, the corresponding solution at time t is

$$\Phi_t(s_0).$$

Thus, if $S(0)$ is the random initial state, then

$$S(t) = \Phi_t(S(0)),$$

and for the deterministic trajectory $\{s(t)\}_{t\geq 0}$ with initial value $s(0)$, we have

$$s(t) = \Phi_t(s(0)).$$

Under the standard continuity and Lipschitz assumptions of the Picard–Lindelöf theorem, the ODE admits a unique solution for each initial condition (Coddington and Levinson, 1955). In one dimension, uniqueness implies that the flow map Φ_t is order-preserving: if $s_0^{(1)} < s_0^{(2)}$, then

$$\Phi_t\big(s_0^{(1)}\big) < \Phi_t\big(s_0^{(2)}\big), \qquad \forall t \geq 0,$$

since otherwise the two trajectories would intersect at some time, contradicting uniqueness.

Because Φ_t is strictly increasing, we have

$$S(t) \leq s(t) \quad \Longleftrightarrow \quad \Phi_t(S(0)) \leq \Phi_t(s(0)) \quad \Longleftrightarrow \quad S(0) \leq s(0).$$

Taking probabilities on both sides yields

$$F_t(s(t)) = \mathbb{P}\big(S(t) \leq s(t)\big) = \mathbb{P}\big(S(0) \leq s(0)\big) = F_0(s(0)).$$

Therefore, the CDF value remains constant along every solution trajectory. □

B Dataset Details

Earthquake (Chen et al., 2021). This dataset records earthquake and aftershock occurrences in Japan between Year 1990 and 2020. Each sequence aggregates one month of events, so the prediction horizon is $T = 30$ days. The processed benchmark contains $1{,}050$ sequences in total, split into 950 training, 50 validation, and 50 test sequences. According to the provided data files, sequence lengths span from 22 to 554 events, with an average of 87.5 events per sequence.

COVID-19. (Chen et al., 2021) The dataset stores daily COVID-19 case events across counties in New Jersey from March 2020 to July 2020. One sequence corresponds to a one-week window, i.e., $T = 7$ days. The full dataset contains $1{,}650$ sequences, partitioned into $1{,}450$ for training, 100 for validation, and 100 for testing. Sequence lengths vary from 5 to 287, and the average length is 97.8.

Citibike. (Yuan et al., 2023) The Citibike benchmark is built from New York City bike-sharing trips collected from April to August 2019. The temporal unit is one hour, and each sequence covers a single day, giving $T = 24$. We use $2{,}440$ training sequences, 300 validation sequences, and 320 test sequences. The processed files contain $3{,}060$ sequences in total, with lengths ranging from 14 to 204 and an average of 134.9 events.

C Training Details and Hyperparameter Settings

For all real-world datasets, we apply a linear shift and scaling to normalize the spatial domain to $[0, 1]^2$ during training. At evaluation time, both the generated event times and spatial locations are mapped back to their original domains before computing the prediction errors.

In our implementation, the temporal and spatial flows share the same Transformer backbone for the encoder and decoder, but use separate MLP heads for velocity prediction. The inter-event time Δt_i is

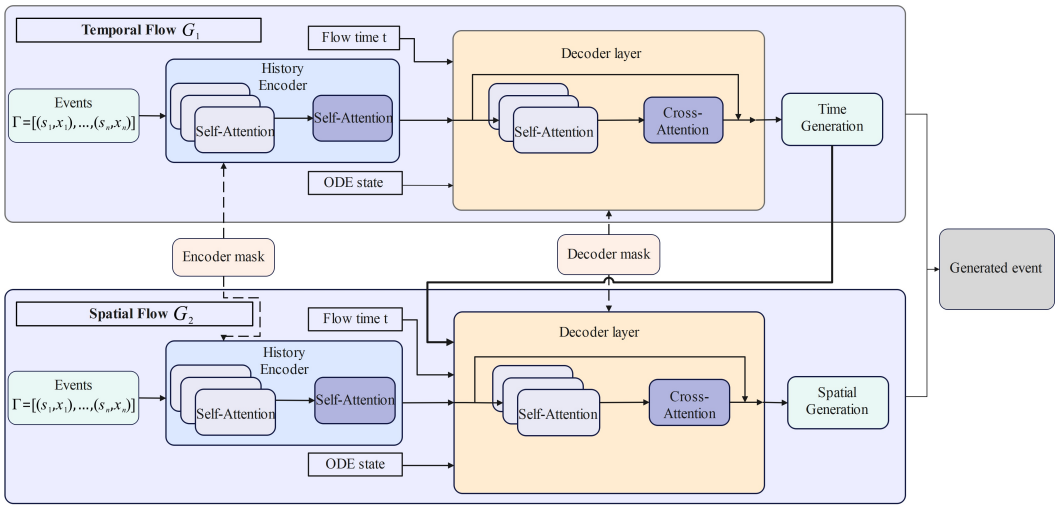


Figure 2: Overview of ARCH framework for spatiotemporal event modeling.

Table 6: Hyperparameter settings of ARCH.

Hyperparameter	Setting
Embedding Dimension in Encoder	32
Embedding Dimension in Decoder	32
Sinusoidal Encoding dimension	32
Input Layer for ODE state of the temporal flow	MLP (64, 32)/GELU
Input Layer for ODE state of the spatial flow	MLP (64, 32)/ GELU
Input Layer for event spatial locations	MLP (64, 32)/GELU
Input Layer for flow time t	Sinusoidal Encoding $\rightarrow$ MLP [32, 32]/GELU
Input Layer for Δs	Log Transform $\rightarrow$ Sinusoidal Encoding
Number of Encoder layers	3
Number of Decoder layers	3
Layer Norm First	True
Dropout	0.15
Feed Forward Dimension in Encoder	32
Feed Forward Dimension in Decoder	32
Number of Attention Heads	1
Temporal Velocity Head	MLP (256, 1)/GELU
Spatial Velocity Head	MLP (256, d)/GELU
Optimizer	AdamW
Learning rate	10^{-3}

passed through a sinusoidal encoding layer to obtain its latent representation. The flow time $t \in [0, 1]$ is first encoded by a sinusoidal encoding layer and then transformed by an MLP. The ODE states and event spatial locations are separately embedded using MLP layers. The relevant representations are then summed to form token embeddings, which are processed by the self-attention layers in the encoder and decoder.

When summing the loss terms, all temporal velocity losses are assigned weight 1, while all spatial velocity losses are assigned weight 1.5. The three mask types are treated equally, i.e., no additional weighting is applied across mask types. For the random-conditioning mask, each binary mask indicator is sampled independently, with probability 0.7 of being one. To sample a consecutive mask, we uniformly sample two endpoints a and b and resample until $a < b$. The architectural hyperparameters of ARCH are listed in Table 6.

All experiments were conducted on a Linux workstation equipped with a NVIDIA GeForce RTX 3090 GPU card (24 GB Memory).

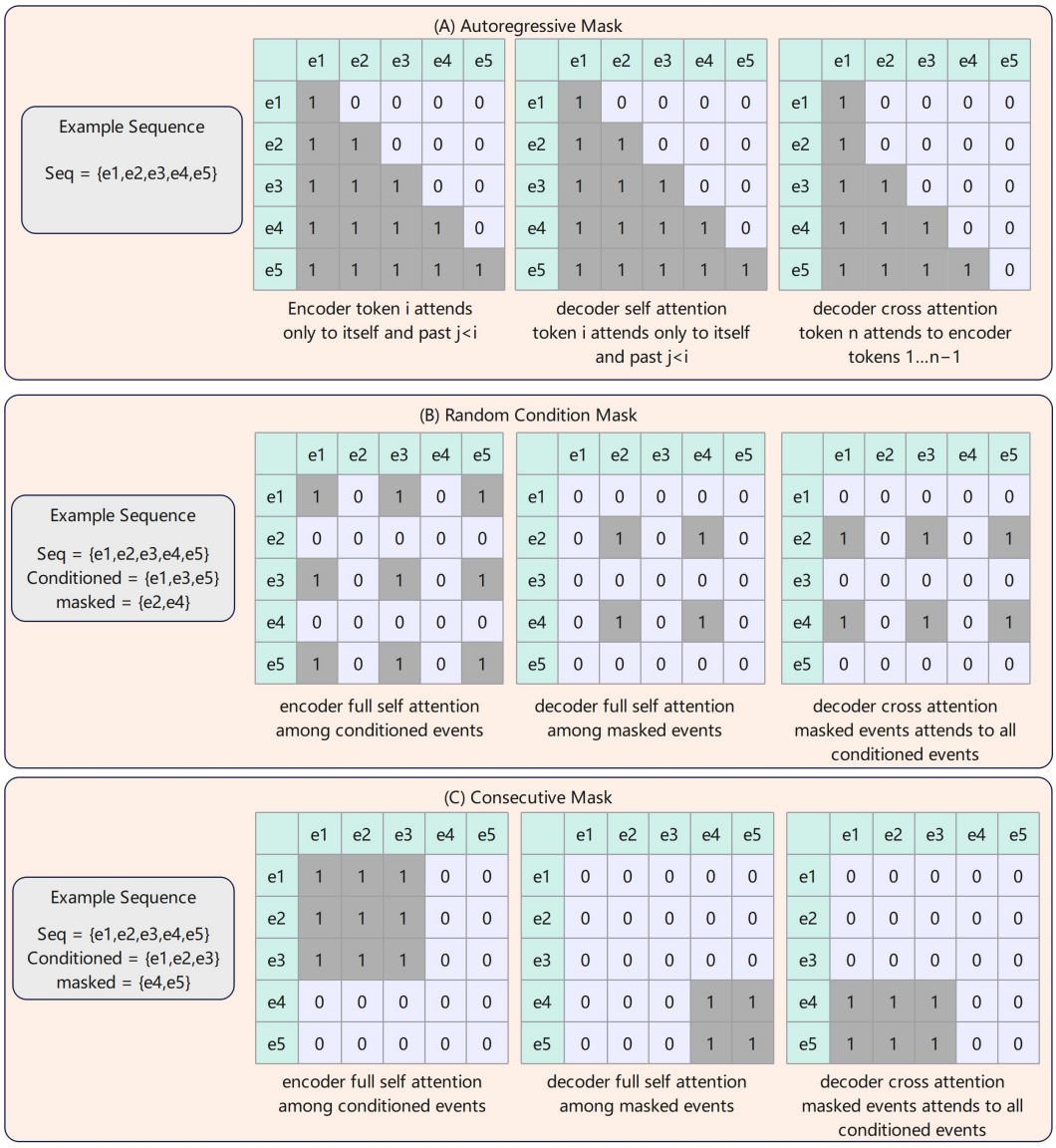


Figure 3: Illustration of the masking strategies used in ARCH training: autoregressive mask, random condition mask, and consecutive mask.

D Test Log-Likelihood

We evaluate held-out probabilistic fit using the training, validation, and test splits from the public Seahorse benchmark (Aalaila et al., 2026). We report the average per-event negative log-likelihood (NLL):

$$\mathrm{NLL} = -\frac{1}{|\mathcal{I}_{\mathrm{test}}|} \sum_{i \in \mathcal{I}_{\mathrm{test}}} \log p(s_i, \mathbf{x}_i \mid \mathcal{H}_i),$$

where $\mathcal{I}_{\mathrm{test}}$ indexes the test events. For ARCH, the joint log-density decomposes into temporal and conditional spatial log-densities. We include baselines with evaluable conditional likelihoods, excluding DSTPP because its original formulation does not admit tractable likelihood evaluation.

As shown in Table 7, ARCH achieves the lowest NLL on COVID-19 and Citibike and the second-lowest on Earthquake, behind DeepSTPP. These results complement the point-prediction evaluation and demonstrate the strong density estimation capabilities of our hierarchical flows for spatiotemporal events.

Algorithm 1 ARCH Training

Require: Training dataset $\mathcal{D}$, temporal flow $\mathbf{v}_{\Theta_1}$, spatial flow $\mathbf{v}_{\Theta_2}$

Apply a log transformation to each inter-event time: $\Delta s_i \mapsto \log(\Delta s_i + \epsilon)$, where $\Delta s_i = s_i - s_{i-1}$ and $\epsilon > 0$ is a small constant.
while not converged **do**
  Sample a mini-batch $\mathcal{B}$ from $\mathcal{D}$ and pad all sequences to the same length.
  For each sequence in $\mathcal{B}$, construct the autoregressive mask $\mathcal{M}_{\mathrm{ar}}$, and sample a random-conditioning mask $\mathcal{M}_{\mathrm{rc}}$ and a consecutive mask $\mathcal{M}_{\mathrm{con}}$.
  for each mask type $\mathcal{M} \in \{\mathcal{M}_{\mathrm{ar}}, \mathcal{M}_{\mathrm{rc}}, \mathcal{M}_{\mathrm{con}}\}$ **do**
    Combine $\mathcal{M}$ with the padding mask to derive the encoder mask $\mathcal{M}^{\mathrm{enc}}$ and decoder mask $\mathcal{M}^{\mathrm{dec}}$ for $\mathcal{B}$.
    Sample initial noise $\mathcal{S}_0$ and $\mathcal{X}_0$ from standard normal distributions for $\mathbf{v}_{\Theta_1}$ and $\mathbf{v}_{\Theta_2}$, respectively.
    Sample flow times $\tau_1, \tau_2 \sim \mathrm{Uniform}(0, 1)$ for the temporal and spatial flows, respectively.
    Construct the interpolated flow states:

$$\mathcal{S}_{\tau_1} = (1 - \tau_1)\mathcal{S}_0 + \tau_1 \mathcal{S}_1, \qquad \mathcal{X}_{\tau_2} = (1 - \tau_2)\mathcal{X}_0 + \tau_2 \mathcal{X}_1.$$

    Apply $(\mathcal{M}^{\mathrm{enc}}, \mathcal{M}^{\mathrm{dec}})$ to $\mathbf{v}_{\Theta_1}$ and $\mathbf{v}_{\Theta_2}$ to obtain velocity predictions $\mathcal{V}_1$ and $\mathcal{V}_2$.
    Compute the masked flow-matching losses:

$$\mathcal{L}_1^{\mathcal{M}} = \left\| \mathcal{M}^{\mathrm{dec}} \circ (\mathcal{V}_1 - (\mathcal{S}_1 - \mathcal{S}_0)) \right\|^2, \qquad \mathcal{L}_2^{\mathcal{M}} = \left\| \mathcal{M}^{\mathrm{dec}} \circ (\mathcal{V}_2 - (\mathcal{X}_1 - \mathcal{X}_0)) \right\|^2.$$

  end for
  Sum all loss terms, backpropagate the gradients, and update the model parameters.
end while

Algorithm 2 ARCH Generation

Require: Observed events, target-event positions, trained temporal flow $\mathbf{v}_{\Theta_1}$ and spatial flow $\mathbf{v}_{\Theta_2}$

Construct the encoder mask $\mathcal{M}^{\mathrm{enc}}$ and decoder mask $\mathcal{M}^{\mathrm{dec}}$ according to the observed and target events.
Sample initial noise $\mathcal{S}_0$ from a standard normal distribution.
Starting from $\mathcal{S}_0$, integrate the temporal ODE from flow time 0 to 1; at each integration step, apply $(\mathcal{M}^{\mathrm{enc}}, \mathcal{M}^{\mathrm{dec}})$ to $\mathbf{v}_{\Theta_1}$ to obtain the velocity field.
Obtain $\mathcal{S}_1$ and map it back to the inter-event-time domain by applying the inverse log transformation, $\Delta s = \exp(\mathcal{S}_1) - \epsilon$.
Sample initial noise $\mathcal{X}_0$ from a standard normal distribution.
Condition the spatial flow $\mathbf{v}_{\Theta_2}$ on $\mathcal{S}_1$. Starting from $\mathcal{X}_0$, integrate the spatial ODE from flow time 0 to 1; at each integration step, apply $(\mathcal{M}^{\mathrm{enc}}, \mathcal{M}^{\mathrm{dec}})$ to $\mathbf{v}_{\Theta_2}$ to obtain the velocity field.
Obtain $\mathcal{X}_1$ as the generated spatial locations of the target events.

E Ablation Studies

E.1 Hierarchical Generation and Flow Matching

To assess the key design choices of ARCH, we considered two variants: (1) Joint Flow, which jointly models event times and locations in a single flow while retaining the same Transformer backbone, masking protocol, and matched parameter count; and (2) M-Transformer, which directly regresses missing event times and locations using the same temporal–spatial factorization, Transformer backbone, and masking protocol. We compare ARCH with Joint Flow on next-event prediction, inverse inference, and internal trajectory recovery, and with M-Transformer on the latter two tasks.

Table 8 reports next-event prediction errors. ARCH achieves lower temporal errors across all three datasets and lower spatial error on COVID-19. The two methods tie on Citibike spatial prediction at the reported precision, while Joint Flow performs slightly better on Earthquake spatial prediction. Overall, ARCH matches or outperforms Joint Flow on five of six metrics.

Table 7: Average per-event test negative log-likelihood (NLL), reported as mean ± standard deviation. Lower is better, and the best results are highlighted in bold.

Model	COVID-19	Earthquake	Citibike
HPP	1.876 ± 0.060	9.021 ± 0.076	-2.837 ± 0.043
STHP	-2.014 ± 0.039	5.462 ± 0.021	-5.720 ± 0.002
NJSDE	-1.799 ± 0.067	5.155 ± 0.160	-5.786 ± 0.001
NSTPP	-1.962 ± 0.125	4.923 ± 0.061	-5.780 ± 0.021
DeepSTPP	-2.104 ± 0.012	$\mathbf{2.705 \pm 0.054}$	-5.256 ± 0.044
ARCH	$\mathbf{-2.152 \pm 0.015}$	4.682 ± 0.040	$\mathbf{-6.019 \pm 0.006}$

Table 8: Next-event prediction errors of hierarchical (ARCH) and joint flows. Entries report means and standard deviations; lower is better.

	Earthquake		COVID-19		Citibike	
Method	Temporal ↓	Spatial ↓	Temporal ↓	Spatial ↓	Temporal ↓	Spatial ↓
Joint Flow	0.382±0.002	**6.54±0.020**	0.098±0.000	0.396±0.002	0.212±0.001	**0.031±0.000**
ARCH	**0.375±0.001**	6.55±0.022	**0.097±0.000**	**0.391±0.000**	**0.211±0.000**	**0.031±0.000**

Table 9 presents inverse inference results. ARCH achieves the lowest errors on both Earthquake metrics and COVID-19 temporal prediction, and ties with Joint Flow on Citibike spatial prediction. Joint Flow performs best on COVID-19 spatial prediction, while M-Transformer leads on Citibike temporal prediction.

Table 10 reports internal trajectory recovery results for contiguous segments of lengths 5 and 10. ARCH achieves the best or tied-best performance on four of six metrics for length 5 and five of six for length 10. M-Transformer achieves the lowest Citibike temporal errors at both lengths, while Joint Flow leads on Earthquake spatial prediction at length 5.

Table 11 reports the results of missing-attribute inference. ARCH achieves the best performance in most settings, with only a few exceptions: Joint Flow performs better in two COVID-19 spatial prediction settings, while M-Transformer achieves slightly lower temporal error on Citibike in two settings.

These results demonstrate that our hierarchical generation preserves competitive predictive accuracy while offering a key computational advantage: explicitly modeling the temporal marginal enables instantaneous risk (conditional intensity) evaluation without spatial marginalization (Eq. (5) and Lemma 3.1). In contrast, Joint Flow does not directly provide this marginal despite its evaluable joint density. Furthermore, compared with direct regression (M-Transformer), our flow-matching formulation achieves competitive or superior prediction accuracy on most metrics while enabling predictive sampling and uncertainty quantification through conditional generative modeling.

E.2 Training Masks

We performed a nested ablation using the autoregressive mask (AR), AR with random conditioning (AR+RC), and the full combination with the consecutive mask (AR+RC+CM). AR is the common base because it defines the next-event objective. This design measures the incremental effect of RC given AR and that of CM given AR+RC; it is not an exhaustive leave-one-mask-out study.

Next-event prediction. Table 12 shows that AR-only training is already competitive. Relative to AR alone, the full mask combination improves Earthquake temporal and COVID-19 spatial prediction, ties on COVID-19 temporal and Citibike spatial prediction, and produces small numerical degradations on Earthquake spatial and Citibike temporal prediction. Thus, the additional conditioning masks do not uniformly improve forward prediction, but preserve broadly comparable accuracy while supporting non-forward tasks.

Scattered partial recovery. Table 13 evaluates randomly scattered missing values at rates of 5% and 10%. Adding RC to AR matches or improves all 12 metrics, with strict numerical improvements in 11 cases. The largest temporal changes occur on Citibike, where the errors decrease from 0.401

Table 9: Inverse inference comparison of hierarchical flows, joint flows, and direct regression. Lower is better. Best means, including ties at the reported precision, are highlighted in bold.

	Earthquake		COVID-19		Citibike	
Model	Spatial ↓	Temporal ↓	Spatial ↓	Temporal ↓	Spatial ↓	Temporal ↓
M-Transformer	7.89±0.030	0.490±0.004	0.466±0.003	0.156±0.002	0.034±0.000	**0.341±0.010**
Joint Flow	7.68±0.019	0.519±0.004	**0.460±0.002**	0.155±0.001	**0.033±0.000**	0.478±0.003
ARCH	**7.60±0.022**	**0.471±0.006**	0.464±0.001	**0.148±0.002**	**0.033±0.000**	0.373±0.011

Table 10: Internal trajectory recovery for contiguous segments of lengths 5 and 10. Lower is better. Best means, including ties at the reported precision, are highlighted in bold.

	Earthquake		COVID-19		Citibike	
Model	Spatial ↓	Temporal ↓	Spatial ↓	Temporal ↓	Spatial ↓	Temporal ↓
Recovery length = 5						
M-Transformer	8.09±0.490	0.465±0.012	0.487±0.004	0.187±0.008	0.032±0.000	**0.161±0.002**
Joint Flow	**7.88±0.113**	0.463±0.010	0.470±0.008	0.177±0.008	**0.031±0.000**	0.184±0.006
ARCH	7.90±0.127	**0.456±0.010**	**0.467±0.007**	**0.173±0.008**	**0.031±0.000**	0.176±0.005
Recovery length = 10						
M-Transformer	7.99±0.033	0.463±0.005	0.482±0.004	0.199±0.005	0.032±0.000	**0.147±0.003**
Joint Flow	7.83±0.074	0.467±0.005	0.466±0.004	0.189±0.004	0.032±0.000	0.162±0.004
ARCH	**7.75±0.060**	**0.458±0.008**	**0.462±0.006**	**0.183±0.006**	**0.031±0.000**	0.156±0.004

to 0.214 and 0.212 at the two rates. Adding CM to AR+RC yields relatively small further changes, indicating that RC provides the primary benefit for scattered completion.

Contiguous trajectory recovery. Table 14 evaluates internal gaps of lengths 5 and 10. Adding RC to AR improves all 12 reported means and ties on the remaining metric. CM then improves four of the six metrics for length 5 and five of the six for length 10, tying on the others. The additional benefit of CM is therefore most apparent for contiguous reconstruction. Together, the results indicate complementary, task-dependent roles: AR provides competitive forward prediction, RC improves scattered completion, and CM further supports contiguous missing segments.

E.3 Encoder–Decoder Architecture

To evaluate the encoder–decoder architecture of ARCH, we compared it with a block-masked decoder-only alternative with a matched parameter count. To enable parallel autoregressive training, the decoder-only model maintains separate copies of the sequence: observed tokens encode conditioning information, while target tokens represent the evolving ODE states.

Table 15 reports next-event prediction errors. The decoder-only model ties with ARCH on Earthquake spatial, COVID-19 temporal, and Citibike spatial prediction, while ARCH achieves lower errors on the remaining three metrics. Overall, ARCH matches or outperforms the decoder-only alternative across all six metrics.

Table 16 compares training costs under identical hardware, batch size, ODE solver settings, and training configurations. The decoder-only model reduces training time per epoch by approximately 35–46% but increases peak GPU memory usage by 33–68%. These results reveal a clear time–memory trade-off: the decoder-only architecture trains faster, whereas our encoder–decoder design is more memory-efficient. The higher memory consumption of the decoder-only model is consistent with its use of separate observed and target sequence copies for parallel autoregressive training.

F Additional Results for Missing Event Recovery

We randomly selected a portion of events. Each selected event has one third chance to miss its time, one third chance to miss its location, and one third chance to be completely missing. We then applied ARCH to predict the missing part of these events. We considered missing ratios of 5% and 10%. We compared with KNNImpute where $k = 10$. As shown in Table 18, ARCH achieves better recover accuracy in all the cases except that on COVID-19 dataset, the event time accuracy of ARCH is

Table 11: Missing event attribute recovery with M-Transformer, Joint Flow, and ARCH. Lower is better. Best means, including ties at the reported precision, are shown in bold.

Model	Earthquake		COVID-19		Citibike	
	Spatial ↓	Temporal ↓	Spatial ↓	Temporal ↓	Spatial ↓	Temporal ↓
Missing ratio = 5%						
M-Transformer	7.25±0.159	0.356±0.037	0.399±0.004	0.092±0.012	**0.031±0.000**	**0.197±0.007**
Joint Flow	6.51±0.023	0.350±0.032	**0.381±0.007**	0.092±0.011	**0.031±0.000**	0.228±0.009
ARCH (Ours)	**6.45±0.163**	**0.348±0.030**	0.383±0.007	**0.090±0.011**	**0.031±0.000**	0.214±0.009
Missing ratio = 10%						
M-Transformer	7.09±0.099	0.372±0.017	0.418±0.002	0.096±0.008	**0.032±0.000**	**0.196±0.005**
Joint Flow	6.32±0.163	0.373±0.018	**0.396±0.003**	0.095±0.008	**0.032±0.000**	0.231±0.007
ARCH (Ours)	**6.26±0.140**	**0.367±0.017**	0.397±0.003	**0.093±0.008**	**0.032±0.000**	0.211±0.006

Table 12: Next-event prediction under nested mask ablations. AR: autoregressive; RC: random conditioning; CM: consecutive masking. Lower is better; entries report means and standard deviations.

Method	Earthquake		COVID-19		Citibike	
	Temporal	Spatial	Temporal	Spatial	Temporal	Spatial
AR	0.385 ± 0.001	6.53 ± 0.015	0.097 ± 0.000	0.392 ± 0.001	0.209 ± 0.001	0.031 ± 0.000
AR+RC	0.381 ± 0.001	6.64 ± 0.028	0.097 ± 0.000	0.391 ± 0.001	0.211 ± 0.001	0.031 ± 0.000
AR+RC+CM	0.375 ± 0.001	6.55 ± 0.022	0.097 ± 0.000	0.391 ± 0.000	0.211 ± 0.000	0.031 ± 0.000

slightly worse than KNNImpute under missing ratio 10%. ARCH also consistently outperforms NHPS in all the cases.

G Visualization of Marginal Temporal Intensity

To complement the spatial intensity snapshots, we visualize the marginal temporal conditional intensity, which describes the instantaneous event rate irrespective of spatial location. For a spatial observation domain $\mathcal{X}$, the marginal temporal conditional intensity is defined as

$$\lambda(s \mid \mathcal{H}_s) = \int_{\mathcal{X}} \lambda(s, \mathbf{x} \mid \mathcal{H}_s)\, d\mathbf{x}.$$

These curves provide a complementary view of how the estimated event risk evolves over time.

Figures 5 and 6 compare ARCH with the ground truth and the competing intensity-based models on a test sequence from each synthetic process. The temporal evaluation uses ten query points per inter-event interval, allowing the curves to reveal intensity evolution between observed events as well as changes associated with event arrivals.

On **STHP-SYN**, the methods broadly reproduce the excitation following event arrivals and the subsequent decay, although their estimated intensity levels differ. On **STSC-SYN**, ARCH follows the characteristic increases between events and decreases following event arrivals, with local deviations from the ground truth. DeepSTPP exhibits extended flat segments that do not reproduce the between-event growth, while AutoSTPP and NSTPP show discrepancies in the shape or magnitude of the temporal intensity. These visualizations complement the spatial snapshots by illustrating the temporal behavior of the recovered event risk.

H Limitations and Future Work

While our framework supports a broad range of inference tasks beyond autoregressive prediction, it currently assumes that the number of target events, or target event attributes, is specified in advance. It does not explicitly infer how many events are missing within an observed time window, or how many future events should occur within a forecasting horizon. Although event counts can in principle be estimated through repeated sampling and rejection, a more principled and efficient approach would be valuable. We leave this direction for future work.

Table 13: Scattered partial missing-event recovery under nested mask ablations. Lower is better; entries report means and standard deviations.

	Earthquake		COVID-19		Citibike	
Method	Temporal	Spatial	Temporal	Spatial	Temporal	Spatial
Missing rate: 5%						
AR	0.377 ± 0.041	6.63 ± 0.184	0.094 ± 0.013	0.392 ± 0.005	0.401 ± 0.037	0.032 ± 0.000
AR+RC	0.355 ± 0.034	6.44 ± 0.171	0.090 ± 0.013	0.383 ± 0.006	0.214 ± 0.008	0.031 ± 0.000
AR+RC+CM	0.348 ± 0.030	6.45 ± 0.163	0.090 ± 0.011	0.383 ± 0.007	0.214 ± 0.009	0.031 ± 0.000
Missing rate: 10%						
AR	0.398 ± 0.016	6.43 ± 0.160	0.099 ± 0.008	0.405 ± 0.003	0.401 ± 0.034	0.032 ± 0.000
AR+RC	0.372 ± 0.016	6.25 ± 0.170	0.094 ± 0.011	0.397 ± 0.002	0.212 ± 0.006	0.032 ± 0.000
AR+RC+CM	0.367 ± 0.017	6.26 ± 0.140	0.093 ± 0.008	0.397 ± 0.003	0.211 ± 0.006	0.032 ± 0.000

Table 14: Contiguous internal-trajectory recovery under nested mask ablations. Lower is better; entries report means and standard deviations.

	Earthquake		COVID-19		Citibike	
Method	Temporal	Spatial	Temporal	Spatial	Temporal	Spatial
Missing length: 5						
AR	0.518 ± 0.014	7.96 ± 0.110	0.189 ± 0.009	0.472 ± 0.006	0.391 ± 0.039	0.032 ± 0.000
AR+RC	0.470 ± 0.005	7.91 ± 0.127	0.177 ± 0.008	0.469 ± 0.007	0.176 ± 0.005	0.031 ± 0.000
AR+RC+CM	0.456 ± 0.010	7.90 ± 0.127	0.173 ± 0.008	0.467 ± 0.007	0.176 ± 0.005	0.031 ± 0.000
Missing length: 10						
AR	0.507 ± 0.005	7.90 ± 0.078	0.201 ± 0.005	0.467 ± 0.006	0.392 ± 0.041	0.032 ± 0.000
AR+RC	0.468 ± 0.009	7.76 ± 0.073	0.188 ± 0.006	0.464 ± 0.004	0.158 ± 0.004	0.031 ± 0.000
AR+RC+CM	0.458 ± 0.008	7.75 ± 0.060	0.183 ± 0.006	0.462 ± 0.006	0.156 ± 0.004	0.031 ± 0.000

In addition, our results demonstrate that flow matching provides a flexible framework for capturing diverse event distributions and their instantaneous risks. However, in many applications, structural priors may be important for uncovering interpretable or semantically meaningful event relationships. Incorporating such priors into the proposed framework is another promising direction for future work. Finally, we plan to extend our approach to support discrete event-mask generation, enabling the model to infer not only the values of target events but also which types of events or attributes should be generated.

Table 15: Next-event prediction for encoder–decoder and decoder-only implementations with comparable parameter counts. Lower is better; entries report means and standard deviations. Best means, including ties at the reported precision, are shown in bold.

	Earthquake		COVID-19		Citibike	
Method	Temporal ↓	Spatial ↓	Temporal ↓	Spatial ↓	Temporal ↓	Spatial ↓
Decoder-only	0.384±0.001	**6.55±0.012**	**0.097±0.000**	0.396±0.001	0.213±0.001	**0.031±0.000**
ARCH	**0.375±0.001**	**6.55±0.022**	**0.097±0.000**	**0.391±0.000**	**0.211±0.000**	**0.031±0.000**

Table 16: Training time per epoch and peak GPU memory for the two architectures. Time is reported in seconds and memory in MB. Lower is better.

	Earthquake		COVID-19		Citibike	
Method	Time	Memory	Time	Memory	Time	Memory
Decoder-only	1.08 s	1,361 MB	1.92 s	1,626 MB	2.61 s	1,052 MB
ARCH	1.92 s	1,025 MB	2.94 s	1,168 MB	4.83 s	628 MB

Table 17: Missing event recovery.

	Earthquake		COVID-19		Citibike	
Model	Spatial ↓	Temporal ↓	Spatial ↓	Temporal ↓	Spatial ↓	Temporal ↓
Missing ratio = 10%						
KNNImp ($k = 3$)	7.28±0.096	0.414±0.012	0.462±0.003	0.096±0.003	0.037±0.000	0.228±0.006
KNNImp ($k = 5$)	7.04±0.075	0.395±0.012	0.441±0.003	0.092±0.004	0.035±0.000	0.221±0.006
KNNImp ($k = 10$)	6.86±0.073	0.381±0.013	0.424±0.002	**0.090±0.004**	0.034±0.000	0.216±0.006
NHPS	N/A	0.418±0.028	N/A	0.093±0.008	N/A	0.220±0.006
ARCH	**6.71±0.022**	**0.365±0.011**	**0.379±0.003**	0.094±0.005	**0.031±0.000**	**0.204±0.005**
Missing ratio = 20%						
KNNImp ($k = 3$)	7.20±0.072	0.418±0.007	0.469±0.003	0.102±0.002	0.037±0.000	0.226±0.004
KNNImp ($k = 5$)	6.95±0.050	0.402±0.007	0.447±0.003	0.099±0.002	0.035±0.000	0.219±0.004
KNNImp ($k = 10$)	6.77±0.049	0.388±0.007	0.429±0.003	**0.097±0.002**	0.034±0.000	0.215±0.004
NHPS	N/A	0.436±0.010	N/A	0.108±0.008	N/A	0.221±0.003
ARCH	**6.57±0.050**	**0.372±0.007**	**0.386±0.003**	0.103±0.002	**0.031±0.000**	**0.198±0.003**
Missing ratio = 30%						
KNNImp ($k = 3$)	7.20±0.028	0.416±0.003	0.469±0.002	0.101±0.001	0.037±0.000	0.228±0.002
KNNImp ($k = 5$)	6.94±0.026	0.398±0.003	0.447±0.002	0.098±0.001	0.036±0.000	0.220±0.002
KNNImp ($k = 10$)	6.76±0.022	0.384±0.004	0.430±0.001	**0.095±0.001**	0.034±0.000	0.215±0.002
NHPS	N/A	0.432±0.017	N/A	0.107±0.003	N/A	0.230±0.006
ARCH	**6.53±0.046**	**0.366±0.006**	**0.385±0.001**	0.099±0.001	**0.031±0.000**	**0.196±0.002**

Table 18: Missing event attribute (time and/or location) recovery. Lower is better. Best means, including ties at the reported precision, are shown in bold.

	Earthquake		COVID-19		Citibike	
Model	Spatial ↓	Temporal ↓	Spatial ↓	Temporal ↓	Spatial ↓	Temporal ↓
Missing ratio = 5%						
KNNImp ($k = 10$)	6.70±0.094	0.374±0.035	0.420±0.003	**0.090±0.010**	0.034±0.000	0.221±0.009
NHPS	N/A	0.393±0.028	N/A	0.099±0.010	N/A	0.215±0.006
ARCH (Ours)	**6.45±0.163**	**0.348±0.030**	**0.383±0.007**	**0.090±0.011**	**0.031±0.000**	**0.214±0.009**
Missing ratio = 10%						
KNNImp ($k = 10$)	6.51±0.051	0.389±0.014	0.431±0.004	**0.089±0.007**	0.034±0.000	0.220±0.005
NHPS	N/A	0.414±0.013	N/A	0.097±0.006	N/A	0.224±0.005
ARCH (Ours)	**6.26±0.140**	**0.367±0.017**	**0.397±0.003**	0.093±0.008	**0.032±0.000**	**0.211±0.006**

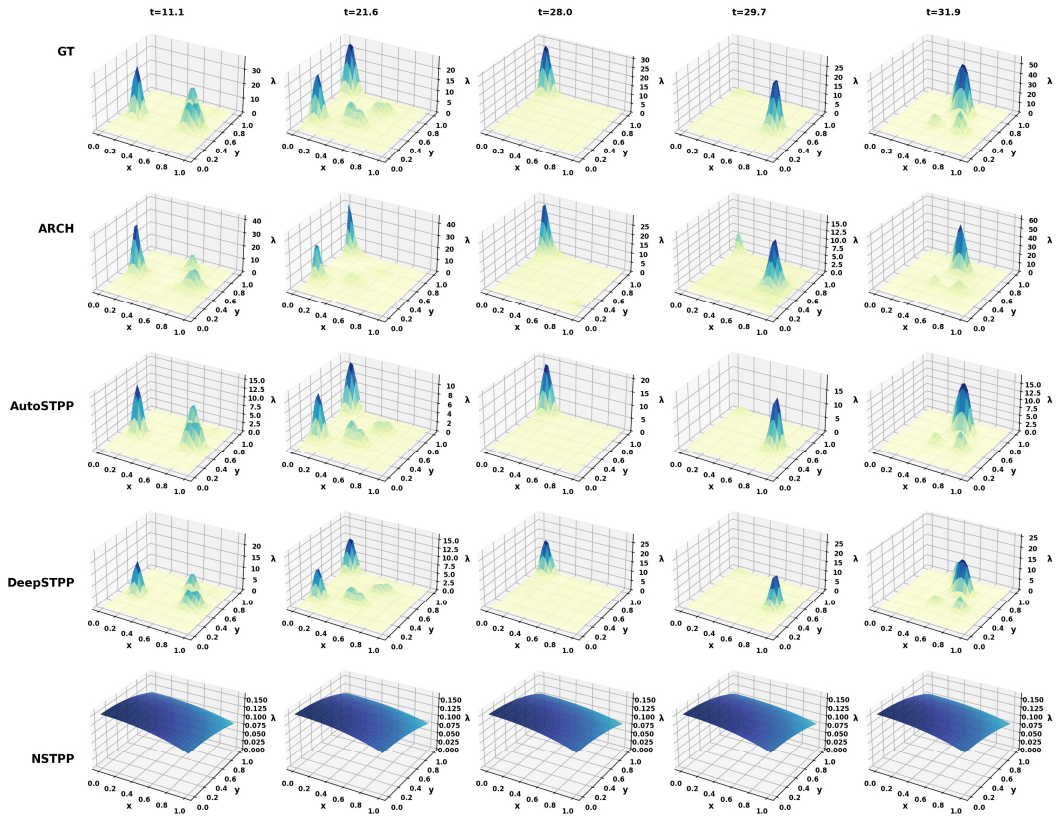


Figure 4: Conditional intensity functions, i.e., instantaneous event risks, at representative time points of a test sequence simulated by a spatiotemporal Hawkes process (**SYN1**). GT denotes the ground truth.

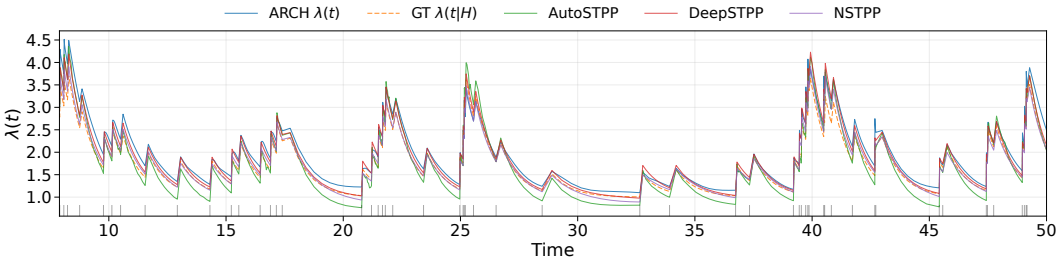


Figure 5: Marginal temporal conditional intensity on a test sequence from **STHP-SYN**, evaluated using ten query points per inter-event interval. GT denotes the ground truth. The vertical tick marks at the bottom indicate observed event times.

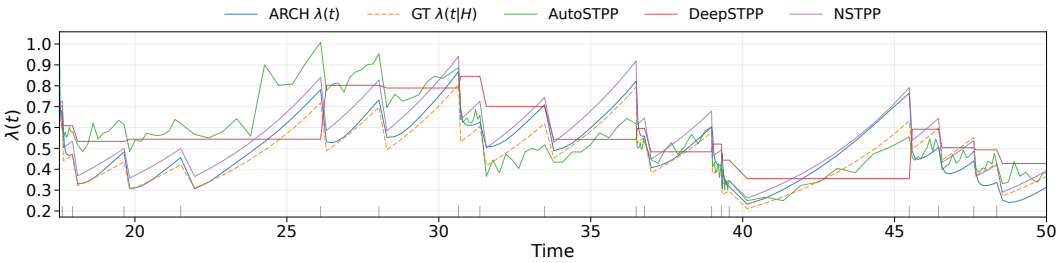


Figure 6: Marginal temporal conditional intensity on a test sequence from **STSC-SYN**, evaluated using ten query points per inter-event interval. GT denotes the ground truth. The vertical tick marks at the bottom indicate observed event times.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We have a clear statement and contribution descriptions in our abstract and introduction.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We have added the limitation discussion in our appendix section.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: We have included our related content in the appendix.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We have included the experiment details in our experiment and appendix sections.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We have attached a code link in our abstract.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: We have described our experiment details in our experiment section and appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We have included the error report and comparison in our results.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We have introduced our computing resources in our appendix section.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We are following the NeurIPS code of ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: There is no societal impact of our work.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: Our research doesn't have any related risks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [N/A]

    Justification: We don't use existing assets.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: We don't release new assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: Our research does not involve human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: Our research does not involve human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: We didn't use LLM for our core method and ideas.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.