AMORTIZED VARIATIONAL INFERENCE OF LATENT INTENSITIES FROM REPLICATED POINT PATTERNS

Anonymous authors
Paper under double-blind review

ABSTRACT

Bayesian inference for latent point process intensities often relies on per-dataset sampling or optimization, making repeated inference computationally expensive. We propose a neural-network-based amortized variational framework that learns a reusable inference rule mapping a variable number of replicated point patterns to a distribution over their shared latent intensity. Trained from event times using the point process likelihood, the model learns an amortized inference rule without ground-truth intensity supervision. At test time, it performs inference of the latent distribution in a forward pass. On synthetic data, additional realizations improve intensity recovery and yield narrower credible intervals with broadly stable coverage. Our method is tested on bicycle-sharing data and achieves the highest mean held-out log likelihood among the evaluated methods while requiring substantially less inference time than the Log-Gaussian Cox Process baseline.

1 INTRODUCTION

Temporal point processes (Daley & Vere-Jones, 2003) provide a general framework for modeling events that occur in time, with applications ranging from earthquakes (Ogata, 1988; 1998) and neural spikes (Amemori & Ishii, 2001) to criminal activity (Mohler et al., 2011). They are commonly characterized by an intensity function that describes how the event rate varies over time. Cox processes (Møller et al., 1998) extend this framework by treating the intensity itself as a latent random function. Bayesian inference quantifies uncertainty over this function from observed events. Inference in this setting includes Markov chain Monte Carlo (MCMC) (Adams et al., 2009), integrated nested Laplace approximations (INLA) (Rue et al., 2009), and variational approximations (Lloyd et al., 2015; Donner & Opper, 2018). These approaches generally compute a posterior distribution per dataset, which can be costly when many distinct data sources must be analyzed.

We consider data consisting of repeated temporal point-pattern observations. A temporal *point pattern* is the sequence of event times observed within a time window: $\mathbb{S}_{\tau,k} = \{t_{\tau,k,1}, \ldots, t_{\tau,k,N_{\tau,k}}\}$.

We group point patterns that are assumed to share the same underlying intensity into a batch and call this a *task*. Thus, a task τ consists of $K_\tau \geq 1$ replicated point patterns, $\mathbb{D}_\tau = \{\mathbb{S}_{\tau,1}, \ldots, \mathbb{S}_{\tau,K_\tau}\}$, on a common observation window $[0, T]$. For example, a task could correspond to neuronal spike trains under one experimental condition, with the task set $\mathbb{D}_\tau$ representing repeated recordings under that condition. Different tasks may correspond to different conditions, such as odor versus visual stimulation, and hence have different underlying firing intensities. Each task has its own unknown latent intensity function $\lambda_\tau : [0, T] \to [0, \infty)$, and the point patterns within the task are modelled as conditionally independent realizations from this shared intensity:

$$\mathbb{S}_{\tau,k} \mid \lambda_\tau \overset{\text{iid}}{\sim} \text{PoissonProcess}(\lambda_\tau), \qquad k = 1, \ldots, K_\tau. \tag{1}$$

The intensity therefore varies across tasks but is shared across realizations within a task. In other words, the K_τ point patterns provide repeated evidence about the same latent function.

Our goal is to design an inference procedure across a population of such tasks. Given the observed point patterns $\mathbb{D}_\tau$ of a new task, we seek to infer its shared latent intensity λ_τ while quantifying uncertainty about the function. Importantly, the number of available realizations K_τ may differ across tasks and may increase as further observations become available. The inference procedure must therefore accept a variable-size, exchangeable set of point patterns and combine the evidence they

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

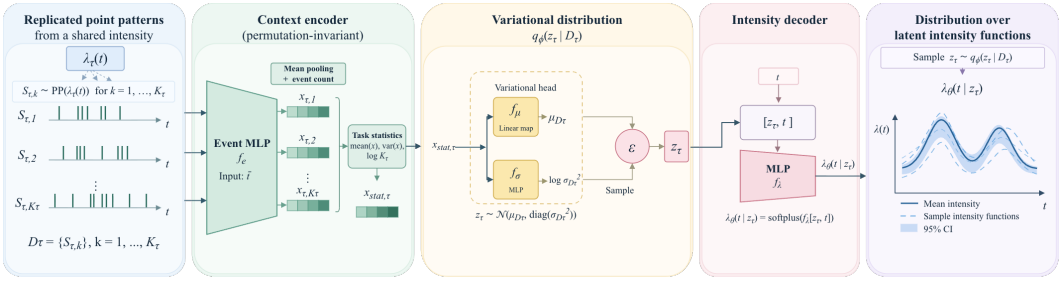


Figure 1: Architecture overview. Replicated point patterns parameterize a variational distribution whose samples are decoded into shared latent intensity functions.

provide about the shared intensity. Drawing on Neural Processes (Garnelo et al., 2018a;b), we learn a single amortized inference rule which can be applied to newly observed tasks (possibly composed of a single point pattern) without per-task optimization. Figure 1 summarizes the framework.

Neural temporal point process models (Du et al., 2016; Omi et al., 2019; Zuo et al., 2020) primarily model history-dependent intensities for event prediction. Recent work extends these predictive models through meta-learning, but retains the same prediction target. Meta Temporal Point Processes (Meta-TPP) (Bae et al., 2023) formulate each event sequence as a separate task and use a context set from that sequence to predict its target event through an NP (Garnelo et al., 2018a;b). Takimoto et al. (2024) encode short sequences into task representations and use them to model intensities for predicting future events. FIM-PP (Berghaus et al., 2026) uses variable-size sets of context sequences and a target history to estimate conditional intensity functions through in-context learning.

More closely related to our setting are methods that treat the intensity itself as the object of inference. Deep Random Splines (DRS) (Loaiza-Ganem et al., 2019) represent each channel's intensity as a random spline. Their DRS-VAE uses separate recurrent encoders to process multiple point patterns. It does not infer a single shared intensity function from exchangeable realizations. Neural Diffusion Intensity Models (NDIM) (Du et al., 2026) model the intensity of a Cox process as a latent path governed by a neural stochastic differential equation (SDE). This formulation assumes that the intensity path is a Markov diffusion and performs amortized variational inference for each observed point pattern, rather than inferring a shared intensity. LGCP-BayesFlow (Wang et al., 2026) amortizes inference for the hyperparameters of Log-Gaussian Cox Process (LGCP) (Møller et al., 1998) models from point patterns, rather than the realized latent intensity function itself. Our focus is amortized distributional inference for one latent intensity shared by a variable number of replicated point patterns. Table 1 compares these methods with our approach.

Our contributions are as follows:

- We develop an amortized framework that maps replicated point patterns to a variational posterior distribution over their shared intensity, using the point process likelihood without ground-truth intensity supervision.
- We empirically demonstrate that the learned inference rule continues to improve intensity recovery and decrease uncertainty as additional realizations become available.
- We evaluate intensity recovery, uncertainty, and inference cost on synthetic data, and held-out model fit on bicycle-sharing data, demonstrating the utility of the learned inference rule without per-dataset optimization.

2 Preliminaries

2.1 Temporal Point Processes

Temporal point processes (Daley & Vere-Jones, 2003) provide a probabilistic framework for modeling event sequences occurring in continuous time. Let $\mathbb{S} = \{t_i\}_{i=1}^{N}$ denote an event sequence observed on a time window $[0, T]$, with counting process $N(t) = \sum_{i=1}^{N} \mathbf{1}\{t_i \leq t\}$ and total

Table 1: Comparisons of related point process methods.

Method	Intensity inference	Intensity UQ	Replicated realizations	Amortized inference
LGCP	✓	✓	✓	×
Meta-TPP	×	×	×	✓
FIM-PP	✓	×	✓	✓
LGCP-BayesFlow	×	×	✓	✓
DRS	✓	✓	×	✓
NDIM	✓	✓	×	✓
Ours	✓	✓	✓	✓

count $N = N(T)$. Let $\mathcal{H}_t$ denote the history of events prior to time t. Its conditional intensity is $\lambda(t \mid \mathcal{H}_t) = \lim_{\Delta t \to 0^+} \mathbb{E}[N(t+\Delta t) - N(t) \mid \mathcal{H}_t]/\Delta t$, the instantaneous event rate given the observed history. The log likelihood of the sequence $\mathbb{S}$ is (Daley & Vere-Jones, 2003):

$$\log p(\mathbb{S} \mid \lambda) = \sum_{i=1}^{N} \log \lambda(t_i \mid \mathcal{H}_{t_i}) - \int_0^T \lambda(t \mid \mathcal{H}_t)\, \mathrm{d}t. \tag{2}$$

A Cox process (Møller et al., 1998) extends this formulation by treating the intensity itself as a latent, non-negative random function λ. Conditional on a realization of λ, the event sequence follows an inhomogeneous Poisson process, $\mathbb{S} \mid \lambda \sim \mathrm{PoissonProcess}(\lambda)$, and its log likelihood is given by Equation 2 thus without history dependence. A prominent instance is the LGCP, which models the log-intensity as a Gaussian process (GP), $f \sim \mathrm{GP}(m, C)$ with $\lambda(t) = \exp\big(f(t)\big)$, where m and C denote the mean and covariance functions.

2.2 Amortized Variational Inference

Variational inference (VI) (Blei et al., 2017) approximates an intractable posterior $p(\mathbf{z} \mid x)$ within a tractable family by minimizing Kullback–Leibler (KL) divergence. Amortized VI (Wu et al., 2020) replaces observation-specific optimization with an inference network $q_\psi(\mathbf{z} \mid x)$ trained across observations $x \sim p_D$:

$$\min_{\psi} \; \mathbb{E}_{x \sim p_D} \left[D_{\mathrm{KL}}(q_\psi(\mathbf{z} \mid x) \,\|\, p(\mathbf{z} \mid x)) \right]. \tag{3}$$

Once trained, the network evaluates $q_\psi(\mathbf{z} \mid x^\star)$ for an unseen observation $x^\star$ in a forward pass.

2.3 Neural Processes

NPs (Garnelo et al., 2018a;b; Dubois et al., 2020) combine stochastic process modeling with amortized neural inference. They are trained across a collection of tasks, where each task consists of observations generated from an underlying function, while different tasks correspond to different functions. By learning similarities across tasks, NPs can adapt to a previously unseen function from a small number of observations in a forward pass.

Within each task, observations are partitioned into a *context set* and a *target set*,

$$\mathbb{C} = \{(x_i, y_i)\}_{i=1}^{N_C}, \qquad \mathbb{T} = \{(x_j, y_j)\}_{j=1}^{N_T}, \tag{4}$$

where $\mathbb{X}_T = \{x_j\}_{j=1}^{N_T}$ and $\mathbb{Y}_T = \{y_j\}_{j=1}^{N_T}$ denote the target inputs and outputs. The context and target sets contain input–output pairs associated with the same underlying function. Given $\mathbb{C}$ and the target inputs $\mathbb{X}_T$, an NP models the conditional predictive distribution $p(\mathbb{Y}_T \mid \mathbb{X}_T, \mathbb{C})$.

Latent NPs introduce a task-level latent variable $\mathbf{z}$ to capture uncertainty about the underlying function (Garnelo et al., 2018b). An encoder maps the context set to a conditional latent distribution $q_\phi(\mathbf{z} \mid \mathbb{C})$, while a decoder generates predictions for the target set conditioned on $\mathbf{z}$. The resulting predictive distribution is

$$p(\mathbb{Y}_T \mid \mathbb{X}_T, \mathbb{C}) = \int p(\mathbb{Y}_T \mid \mathbb{X}_T, \mathbf{z})\, q_\phi(\mathbf{z} \mid \mathbb{C})\, \mathrm{d}\mathbf{z}, \quad p(\mathbb{Y}_T \mid \mathbb{X}_T, \mathbf{z}) = \prod_{j=1}^{N_T} p(y_j \mid x_j, \mathbf{z}). \tag{5}$$

3 METHOD

3.1 PROBLEM FORMULATION

We now extend the setup from Section 1 to a hierarchical model where each task $\tau \sim p(\tau)$ is characterized by an unknown nonnegative intensity function $\lambda_\tau : [0, T] \to [0, \infty)$, and is observed through a dataset $\mathbb{D}_\tau$ of $K_\tau \geq 1$ iid realizations:

$$\mathbb{D}_\tau = \{\mathbb{S}_{\tau,k}\}_{k=1}^{K_\tau}, \qquad \mathbb{S}_{\tau,k} = \{t_{\tau,k,i}\}_{i=1}^{N_{\tau,k}} \mid \lambda_\tau \overset{\text{iid}}{\sim} \text{PoissonProcess}(\lambda_\tau), \tag{6}$$

with $t_{\tau,k,i} \in [0, T]$ denoting the i-th event time in the k-th sequence with event count $N_{\tau,k}$.

Given M training tasks $\{\mathbb{D}_\tau\}_{\tau=1}^M$, our goal is to learn an amortized inference rule $\mathcal{A}$ that maps each task to a variational distribution over its shared intensity function, $\mathcal{A} : \mathbb{D}_\tau \mapsto \mathcal{Q}(\lambda_\tau \mid \mathbb{D}_\tau)$. The rule $\mathcal{A}$ is shared across tasks and accepts a variable number of replications K_τ. Consequently, for a new task $\tau^\star$, inference is performed in a forward pass.

3.2 MODEL OVERVIEW

We represent each task by a latent variable $\mathbf{z}_\tau \in \mathbb{R}^{d_z}$ that captures information about its unknown intensity λ_τ. A decoder parameterized by θ maps $\mathbf{z}_\tau$ to the intensity $\lambda_\tau(t) = \lambda_\theta(t \mid \mathbf{z}_\tau)$.

Since conditionally on the shared intensity λ_τ, the K_τ realizations are independent, using Equation 2, their joint likelihood is

$$p_\theta(\mathbb{D}_\tau \mid \mathbf{z}_\tau) = \exp\left(-K_\tau \int_0^T \lambda_\theta(t \mid \mathbf{z}_\tau)\, \mathrm{d}t\right) \prod_{k=1}^{K_\tau} \prod_{i=1}^{N_{\tau,k}} \lambda_\theta(t_{\tau,k,i} \mid \mathbf{z}_\tau). \tag{7}$$

Given the observed realizations $\mathbb{D}_\tau$, a context encoder parameterized by ϕ defines a learned latent distribution $q_\phi(\mathbf{z}_\tau \mid \mathbb{D}_\tau)$. The encoder–decoder combination induces a variational distribution over the shared intensity of task τ, $q_{\phi,\theta}(\lambda_\tau \mid \mathbb{D}_\tau)$, as defined in Equation 14. We train the context encoder and decoder using the context–target conditional objective in Section 3.5. We next describe the context encoder, decoder, and training objective.

3.3 CONTEXT ENCODER

The context encoder maps the observed realizations $\mathbb{D}_\tau$ to the latent distribution $q_\phi(\mathbf{z}_\tau \mid \mathbb{D}_\tau)$ via aggregation over events and realizations.

Each event $t_{\tau,k,i}$ is standardized as $\bar{t}_{\tau,k,i} \in \mathbb{R}$ using training statistics and embedded via an MLP f_e:

$$\boldsymbol{h}_{\tau,k,i} = f_e(\bar{t}_{\tau,k,i}) \in \mathbb{R}^{d_h}. \tag{8}$$

We perform mean pooling across events within each realization $\mathbb{S}_{\tau,k}$ to obtain the realization embedding $\boldsymbol{u}_{\tau,k}$, and form the augmented realization vector $\boldsymbol{x}_{\tau,k}$:

$$\boldsymbol{u}_{\tau,k} = \frac{1}{N_{\tau,k}} \sum_{i=1}^{N_{\tau,k}} \boldsymbol{h}_{\tau,k,i}, \qquad \boldsymbol{x}_{\tau,k} = [\boldsymbol{u}_{\tau,k}^\top, \log(1 + N_{\tau,k})]^\top \in \mathbb{R}^{d_h+1}, \tag{9}$$

Mean pooling ensures permutation invariance within each realization, while $\log(1 + N_{\tau,k})$ retains information about its event count. To quantify variation across realizations and adapt the size of uncertainty sets to K_τ, we augment the realization vectors with information about the marginal variance across realizations and the number of replicated sequences:

$$\boldsymbol{x}_{\text{stat},\tau} = [\text{mean}_k(\boldsymbol{x}_{\tau,k}),\ \text{var}_k(\boldsymbol{x}_{\tau,k}),\ \log K_\tau] \in \mathbb{R}^{2(d_h+1)+1}, \tag{10}$$

where the variance is computed coordinate-wise across realizations. The mean and log-variance heads both operate on $\boldsymbol{x}_{\text{stat},\tau}$:

$$\boldsymbol{\mu}_{\mathbb{D}_\tau} = f_\mu(\boldsymbol{x}_{\text{stat},\tau}), \qquad \boldsymbol{\sigma}^2_{\mathbb{D}_\tau} = \exp(f_\sigma(\boldsymbol{x}_{\text{stat},\tau}))\,, \tag{11}$$

Here, f_μ is a linear map and f_σ is an MLP parameterizing the latent mean and log-variance, respectively. These heads yield the diagonal Gaussian variational posterior as in a Variational Autoencoder (VAE) (Kingma & Welling, 2014),

$$q_\phi(\mathbf{z}_\tau \mid \mathbb{D}_\tau) = \mathcal{N}\big(\boldsymbol{\mu}_{\mathbb{D}_\tau}, \text{diag}(\boldsymbol{\sigma}^2_{\mathbb{D}_\tau})\big)\,. \tag{12}$$

Architecture and standardization details are provided in Appendix A.1.

3.4 Intensity Decoder

The decoder maps the latent variable $\mathbf{z}_\tau$ and query time t to a continuous-time intensity using an MLP f_λ with a softplus (Zhang et al., 2020; Sulem et al., 2024; Zuo et al., 2020) output:

$$\lambda_\tau(t) = \lambda_\theta(t \mid \mathbf{z}_\tau) = \mathrm{softplus}(f_\lambda[\mathbf{z}_\tau, t])\,. \tag{13}$$

The latent distribution therefore induces a distribution over intensity functions,

$$q_{\phi,\theta}(\lambda_\tau \mid \mathbb{D}_\tau) = \int p_\theta(\lambda_\tau \mid \mathbf{z}_\tau) q_\phi(\mathbf{z}_\tau \mid \mathbb{D}_\tau)\, \mathrm{d}\mathbf{z}_\tau. \tag{14}$$

Decoder architecture is detailed in Appendix A.2.

3.5 Variational Inference

For comparison, standard VI for a latent variable model with a fixed prior $p(\mathbf{z}_\tau) = \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ optimizes an evidence lower bound (ELBO) over the complete dataset $\mathbb{D}_\tau$,

$$\log p_\theta(\mathbb{D}_\tau) \geq \mathbb{E}_{q_\phi(\mathbf{z}_\tau \mid \mathbb{D}_\tau)}[\log p_\theta(\mathbb{D}_\tau \mid \mathbf{z}_\tau)] - D_{\mathrm{KL}}[q_\phi(\mathbf{z}_\tau \mid \mathbb{D}_\tau) \,\|\, p(\mathbf{z}_\tau)]\,. \tag{15}$$

We train a conditional model to predict held-out realizations from an observed context. Following the context–target construction of NPs (Garnelo et al., 2018b), we partition each training task into disjoint context and target sets,

$$\mathbb{C}_\tau = \{\mathbb{S}_{\tau,k}\}_{k=1}^{K_C}, \qquad \mathbb{T}_\tau = \{\mathbb{S}_{\tau,k}\}_{k=K_C+1}^{K_\tau}, \qquad \mathbb{C}_\tau \cup \mathbb{T}_\tau = \mathbb{D}_\tau, \qquad K_\tau = K_C + K_T. \tag{16}$$

While inference at test time accommodates any $K \geq 1$, training requires $K_\tau \geq 2$ to ensure non-empty context and target subsets.

Given context $\mathbb{C}_\tau$, the encoder defines a context-conditioned latent distribution $q_\phi(\mathbf{z}_\tau \mid \mathbb{C}_\tau)$. Following the NP construction, we use this distribution as a context-conditioned prior, inducing the conditional model

$$\widetilde{p}_{\phi,\theta}(\mathbb{T}_\tau \mid \mathbb{C}_\tau) = \int p_\theta(\mathbb{T}_\tau \mid \mathbf{z}_\tau) q_\phi(\mathbf{z}_\tau \mid \mathbb{C}_\tau)\, \mathrm{d}\mathbf{z}_\tau. \tag{17}$$

The latent posterior under this conditional model is

$$\widetilde{p}_{\phi,\theta}(\mathbf{z}_\tau \mid \mathbb{T}_\tau, \mathbb{C}_\tau) = \frac{p_\theta(\mathbb{T}_\tau \mid \mathbf{z}_\tau) q_\phi(\mathbf{z}_\tau \mid \mathbb{C}_\tau)}{\widetilde{p}_{\phi,\theta}(\mathbb{T}_\tau \mid \mathbb{C}_\tau)}. \tag{18}$$

During training, $q_\phi(\mathbf{z}_\tau \mid \mathbb{C}_\tau \cup \mathbb{T}_\tau)$ serves as a variational approximation to this posterior, giving the conditional ELBO

$$\log \widetilde{p}_{\phi,\theta}(\mathbb{T}_\tau \mid \mathbb{C}_\tau) \geq \mathbb{E}_{q_\phi(\mathbf{z}_\tau \mid \mathbb{C}_\tau \cup \mathbb{T}_\tau)}[\log p_\theta(\mathbb{T}_\tau \mid \mathbf{z}_\tau)] - D_{\mathrm{KL}}[q_\phi(\mathbf{z}_\tau \mid \mathbb{C}_\tau \cup \mathbb{T}_\tau) \,\|\, q_\phi(\mathbf{z}_\tau \mid \mathbb{C}_\tau)]\,. \tag{19}$$

Conditional on $\mathbf{z}_\tau$, the target realizations are independent, with conditional log likelihood

$$\log p_\theta(\mathbb{T}_\tau \mid \mathbf{z}_\tau) = \sum_{\mathbb{S} \in \mathbb{T}_\tau} \left[\sum_{t_i \in \mathbb{S}} \log \lambda_\theta(t_i \mid \mathbf{z}_\tau) - \int_0^T \lambda_\theta(t \mid \mathbf{z}_\tau)\, \mathrm{d}t \right]. \tag{20}$$

The encoder, and hence the resulting conditional model, is permutation invariant with respect to the ordering of replicated point patterns. A derivation of the conditional objective is provided in Appendix B.

4 Experiments

We test whether amortized inference can recover shared latent intensities without per-task fitting, and whether recovery and uncertainty adapt as additional realizations become available.

We evaluate the proposed method along three dimensions: (i) recovery and uncertainty quantification for latent intensity functions, (ii) adaptation as additional replicated point patterns become available, and (iii) computational efficiency relative to per-task inference. We first study these properties in a controlled synthetic setting, where the latent intensity is known, and then evaluate held-out model fit on real bicycle-sharing data. We approximate the likelihood integral over $[0, T]$ using the trapezoidal rule on $G = 300$ equally spaced points and estimate the expectation in Equation 19 with $m = 3$ reparameterized latent samples per task.

4.1 Synthetic Experiments

Synthetic experiments allow us to evaluate inference against known latent intensity functions. We assess whether the proposed amortized inference procedure accurately recovers these intensities and quantifies uncertainty, and whether both intensity recovery and uncertainty adapt as additional replicated point patterns become available. We further examine whether this evidence accumulation extends to numbers of realizations beyond those encountered during training.

4.1.1 Experimental Setup

Data generation and protocol. For each task τ, we independently draw a latent function on $[0, 10]$

$$f_\tau \sim \mathrm{GP}(0, C_{\mathrm{RBF}}), \qquad \lambda_\tau(t) = 20\,\mathrm{softplus}(f_\tau(t)), \tag{21}$$

where $C_{\mathrm{RBF}}(t, t') = \sigma_f^2 \exp(-(t-t')^2/(2\ell^2))$ with signal variance $\sigma_f^2 = 1$ and lengthscale $\ell = 2$. Conditionally on λ_τ, the realizations within each task are generated independently from the corresponding Poisson process. Thus, each synthetic task has a distinct latent intensity, while its replicated point patterns provide repeated observations of that same intensity.

We use 4,500 training, 500 validation, and 500 test tasks, denoted by $\mathcal{D}_{\mathrm{train}}$, $\mathcal{D}_{\mathrm{val}}$, and $\mathcal{D}_{\mathrm{test}}$, respectively. Neither training nor checkpoint selection uses ground-truth intensity labels.

For our method, we select checkpoints based on held-out model fit: how well the inferred intensity distribution explains realizations excluded from its context. For each task $\mathbb{D}_\tau$, we hold out each realization $\mathbb{S}_{\tau,h}$ in turn and use the remaining realizations $\mathbb{D}_{\tau,-h}$ as context. For $s \in \{\mathrm{val}, \mathrm{test}\}$, we define the held-out log likelihood (HLL) as

$$\mathrm{HLL}_s = \frac{1}{|\mathcal{D}_s|} \sum_{\mathbb{D}_\tau \in \mathcal{D}_s} \frac{1}{|\mathbb{D}_\tau|} \sum_{h=1}^{|\mathbb{D}_\tau|} \log \mathbb{E}_{q_{\phi,\theta}(\lambda_\tau \mid \mathbb{D}_{\tau,-h})} \left[p(\mathbb{S}_{\tau,h} \mid \lambda_\tau) \right]. \tag{22}$$

The expectation is approximated using 300 latent samples. After training, we compute the HLL of each saved checkpoint on the validation tasks and select the checkpoint with the highest validation HLL separately for each training seed. Further training and selection details are provided in Appendix C.2.

To study the effect of the number of training realizations, we fix $K_\tau = K_{\mathrm{train}}$ for all training tasks and sample the context size as $K_C \sim \frac{1}{2}\mathrm{Unif}\{\lceil K_{\mathrm{train}}/2 \rceil, \ldots, K_{\mathrm{train}} - 1\} + \frac{1}{2}\mathrm{Unif}\{1, \ldots, K_{\mathrm{train}} - 1\}$, with the target size given by $K_T = K_{\mathrm{train}} - K_C$, where Unif denotes the discrete uniform distribution over the specified set. This sampling scheme favors $K_C \geq K_T$ on average to limit the information gap between the context and full variational distributions.

We train models with $K_{\mathrm{train}} \in \{4, 8, 16\}$. At test time, we progressively increase the number K_{test} of realizations supplied to the encoder, using nested subsets from the same task. This allows us to assess whether a fixed trained model continues to accumulate evidence as additional realizations become available, including when $K_{\mathrm{test}} > K_{\mathrm{train}}$. All configurations are trained for 20,000 updates with four random seeds. Dataset statistics and further experimental details are provided in Appendices C.1 and C.2.

4.1.2 Baselines and Metrics

Baselines. To evaluate recovery and uncertainty, we compare against two classical nonparametric intensity estimators, a Bayesian task-specific estimator and an amortized neural method: (i) **PSTH** (Kass et al., 2003), a peristimulus time histogram with bin width selected on the validation set; (ii) **KDE** (Diggle, 1985), a kernel estimator with bandwidth selected on the validation set; (iii) **LGCP**, a GP intensity model with an exponential link fitted to each task using sparse variational inference; and (iv) **FIM-PP** (Berghaus et al., 2026), a pretrained model designed for conditional intensity prediction. LGCP additionally provides uncertainty estimates, whereas PSTH, KDE, and FIM-PP provide point estimates only. We do not include NDIM (Du et al., 2026), as no public implementation was available at the time of our experiments.

Metrics. We report intensity **RMSE** using the variational posterior mean, pointwise 95% credible interval (CI) coverage (**C95**), mean 95% CI width (**IW95**), and per-task runtime. C95 and IW95

apply only to methods with uncertainty estimates. For our method and LGCP, RMSE and interval metrics are estimated using 300 posterior intensity samples. Details are given in Appendices C.3 and C.5.

4.1.3 RESULTS

Recovery and uncertainty. Table 2 shows that our method achieves the lowest RMSE (1.55), compared with 1.86 for LGCP and 1.91 for KDE. Among the methods that quantify uncertainty, our method attains empirical 95% coverage of 93.28%, compared with 92.70% for LGCP, while producing narrower credible intervals (5.73 versus 6.67). Thus, the reduction in interval width is accompanied by comparable empirical coverage, although both methods remain below the nominal 95% level.

Accuracy–efficiency trade-off. Once trained, our method performs inference for a new task without task-specific optimization, requiring 7.53 ms per task compared with 6.31 s for LGCP. The classical PSTH and KDE estimators are faster, requiring 0.04 ms and 0.75 ms, respectively, but yield higher RMSE and do not provide uncertainty estimates. Our method therefore combines low inference cost with accurate intensity recovery and uncertainty quantification.

Table 2: Synthetic results on 500 test tasks with $K_{\text{test}} = 4$. C95 is reported as a percentage (%). For learned models, $\pm$ denotes standard deviation across four training seeds on the same test set. Classical baseline metrics are deterministic.

	Recovery	**Uncertainty**		**Efficiency**
Method	**RMSE** $\downarrow$	**C95 (%)**	**IW95**	**Time (ms)** $\downarrow$
FIM-PP	14.19	—	—	5.81
PSTH	2.45	—	—	0.04
KDE	1.91	—	—	0.75
LGCP	$1.86_{\pm 0.01}$	$92.70_{\pm 0.10}$	$6.67_{\pm 0.01}$	$6{,}311.76_{\pm 36.06}$
Ours ($K_{\text{train}} = 4$)	$\mathbf{1.55}_{\pm 0.01}$	$93.28_{\pm 0.18}$	$5.73_{\pm 0.00}$	$7.53_{\pm 0.44}$

Evidence accumulation. Table 3 shows that the model trained with $K_{\text{train}} = 4$ benefits from additional realizations. Increasing K_{test} from 4 to 64 reduces RMSE from 1.55 to 0.82 and the mean 95% interval width from 5.73 to 3.74, while coverage remains similar (93.28% and 93.16%, respectively). Figure 2 illustrates this behavior: as additional realizations from the same task are observed, the posterior mean approaches the true intensity and the credible intervals become narrower. Results for $K_{\text{train}} \in \{8, 16\}$ are reported in Appendix C.4. Additional out-of-distribution evaluations are reported in Appendix C.6.

Table 3: Evidence accumulation for the model trained with $K_{\text{train}} = 4$. Results are averaged over four seeds.

K_{test}	**RMSE** $\downarrow$	**C95 (%)**	**IW95**
1	$2.68_{\pm 0.01}$	$92.21_{\pm 0.05}$	$9.36_{\pm 0.02}$
2	$1.99_{\pm 0.00}$	$93.18_{\pm 0.09}$	$7.20_{\pm 0.02}$
4	$1.55_{\pm 0.01}$	$93.28_{\pm 0.18}$	$5.73_{\pm 0.00}$
8	$1.23_{\pm 0.01}$	$93.78_{\pm 0.25}$	$4.78_{\pm 0.01}$
16	$1.00_{\pm 0.01}$	$94.32_{\pm 0.28}$	$4.23_{\pm 0.01}$
32	$0.88_{\pm 0.01}$	$93.84_{\pm 0.21}$	$3.91_{\pm 0.02}$
64	$0.82_{\pm 0.02}$	$93.16_{\pm 0.30}$	$3.74_{\pm 0.03}$

4.2 REAL-DATA EXPERIMENTS

We use the dataset of Divvy bicycle-share trip records from Chicago, USA, spanning 2013–2019.[1] Each trip start is treated as an event, and a point pattern consists of the trip start times observed at

[1] City of Chicago, *Divvy Trips*: https://data.cityofchicago.org/d/fg6s-gzvg.

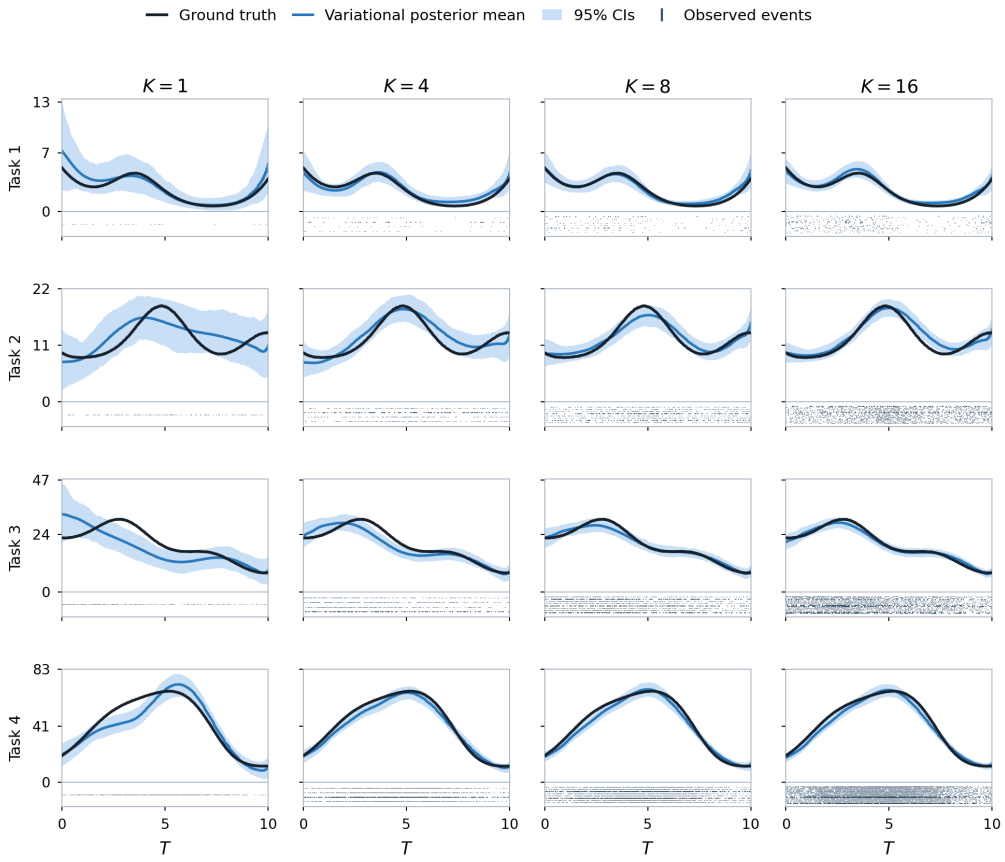


Figure 2: Adaptation to additional realizations for the model trained with $K_{\text{train}} = 8$. Four test tasks are shown as $K_{\text{test}} \in \{1, 4, 8, 16\}$ realizations become available. Across these examples, additional realizations generally improve intensity recovery and yield narrower CIs.

one station on a given day. To construct replicated observations, we group four days matching in station, year, month, and weekday into a task. Thus, each task contains $K = 4$ point patterns that are treated as approximate realizations of a shared daily intensity. Calendar matching and additional count-homogeneity screening are used to make this shared intensity assumption more plausible. Event times are rescaled to map each day to $[0, 10]$.

After screening and subsampling, we obtain train/validation/test splits containing 5,387/674/674 tasks from 182/21/21 stations (224 total). Stations are assigned disjointly across the three splits, so that all test tasks correspond to stations that were unseen during training. Further preprocessing and screening details are given in Appendix D.1.

We report HLL as defined in Equation 22. LGCP follows the same evaluation protocol, whereas PSTH, KDE, and FIM-PP use their point intensity estimates. Further details are provided in Appendices C.2 and D.3.

We train our model on $\mathcal{D}_{\text{train}}$ for 20,000 updates. After training, we evaluate all saved checkpoints on $\mathcal{D}_{\text{val}}$ using the same four-fold HLL with 300 latent samples and select the checkpoint with the highest validation HLL separately for each training seed. The validation set is also used to select the PSTH bin width and KDE bandwidth. Training details are provided in Appendix D.2. Table 4 reports the test-set results.

On tasks from previously unseen stations, our model infers a distribution over the shared intensity from three replicated point patterns. It achieves the highest mean HLL among the evaluated methods (Table 4), outperforming LGCP, which requires per-task optimization. Mean computation time is

26.82 ms per task, compared with 24.4 seconds for LGCP. On unseen stations, the learned inference rule therefore combines strong held-out model fit with substantially lower computational cost.

Figure 3 provides a qualitative held-out predictive check. All curves are inferred using only the three context realizations shown in the lower, light rows; the upper, dark row is a fourth realization used only for evaluation. All methods recover the two main periods of elevated activity, around $t \approx 3.5$ and $t \approx 7$. Our inferred mean preserves a low-intensity region between the peaks and produces a more localized second peak, where the held-out events also concentrate.

Table 4: Divvy results (held-out log likelihood and time) on 674 test tasks. $\pm$ denotes standard deviation across four seeds for Ours and LGCP.

Method	**HLL** $\uparrow$	**Time (ms)** $\downarrow$
FIM-PP	59.53	76.03
PSTH	102.73	**0.19**
KDE	107.10	10.41
LGCP	$108.80_{\pm 0.01}$	$24{,}407.47_{\pm 76.83}$
Ours	$\mathbf{110.99}_{\pm 0.03}$	$26.82_{\pm 0.13}$

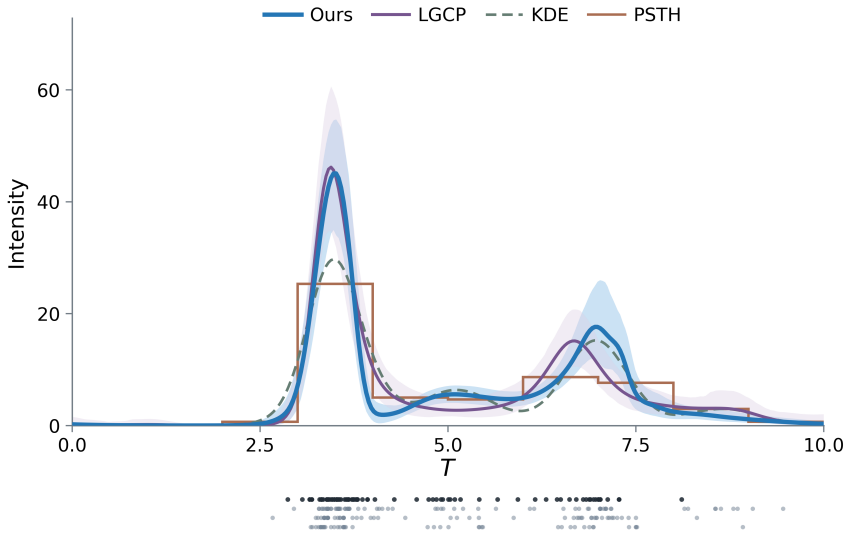


Figure 3: Held-out model fit for a test task. All estimates use only the three context realizations shown in the lower, light rows. The upper, dark row shows the held-out realization and is used only for evaluation. Solid curves show the inferred mean intensities for Ours and LGCP, with 95% CIs; the dashed and step curves show the KDE and PSTH estimates, respectively.

5 Conclusion

We presented an amortized variational framework for inferring a shared latent intensity from replicated point patterns using the point process likelihood. The learned inference rule improves intensity recovery and narrows credible intervals as additional realizations become available, including beyond the training replication count, with broadly stable empirical coverage in this extrapolation regime. On bicycle-sharing data, it achieves the highest mean held-out log likelihood among the evaluated methods, while requiring substantially less inference time than LGCP. Together, these results show that a single trained model can refine intensity estimates and their uncertainty as evidence accumulates, without per-task optimization. Quantifying and reducing the amortization gap is an important direction for future work.

AI USE STATEMENT

In this work, we used generative AI tools to assist with implementing the proposed method, data cleaning, and interpretation of results. We also used these tools to assist with generating initial drafts of parts of the manuscript, polishing the writing, editing code, searching for and summarizing relevant literature, checking references, formatting LaTeX, and preparing Figure 1. All AI-assisted outputs were reviewed and verified by the authors, who take full responsibility for the final content of the manuscript.

REPRODUCIBILITY STATEMENT

We provide anonymized code and data for our proposed method in an anonymous repository. The repository includes scripts for model training and evaluation, the configurations and random seeds used for the reported experiments with our method, as well as scripts for producing its main evaluation results. We also provide the trained checkpoints used for evaluation.

REFERENCES

Ryan Prescott Adams, Iain Murray, and David J. C. MacKay. Tractable nonparametric bayesian inference in poisson processes with gaussian process intensities. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 9–16, 2009. doi: 10.1145/1553374.1553376. URL https://doi.org/10.1145/1553374.1553376.

Ken-ichi Amemori and Shin Ishii. Gaussian process approach to spiking neurons for inhomogeneous Poisson inputs. *Neural Computation*, 13(12):2763–2797, 2001. doi: 10.1162/089976601317098529.

Wonho Bae, Mohamed Osama Ahmed, Frederick Tung, and Gabriel L. Oliveira. Meta Temporal Point Processes. In *International Conference on Learning Representations*, 2023.

David Berghaus, Patrick Seifner, Kostadin Cvejoski, César Ali Ojeda Marin, and Ramses Sanchez. In-context learning of temporal point processes with foundation inference models. In *International Conference on Learning Representations*, pp. 29610–29641, 2026.

David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877, 2017. doi: 10.1080/01621459.2017.1285773.

D. J. Daley and D. Vere-Jones. *An Introduction to the Theory of Point Processes: Volume I: Elementary Theory and Methods*. Springer, 2003. doi: 10.1007/b97277.

Peter Diggle. A kernel method for smoothing point process data. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 34(2):138–147, 1985. doi: 10.2307/2347366.

Christian Donner and Manfred Opper. Efficient Bayesian inference of sigmoidal Gaussian Cox processes. *Journal of Machine Learning Research*, 19(67):1–34, 2018.

Nan Du, Hanjun Dai, Rakshit Trivedi, Utkarsh Upadhyay, Manuel Gomez-Rodriguez, and Le Song. Recurrent marked temporal point processes: Embedding event history to vector. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1555–1564, 2016. doi: 10.1145/2939672.2939875.

Xinlong Du, Harsha Honnappa, and Vinayak Rao. Neural diffusion intensity models for point process data. In *Proceedings of the 42nd Conference on Uncertainty in Artificial Intelligence*, volume 337 of *Proceedings of Machine Learning Research*, pp. 1448–1473. PMLR, 2026. URL https://proceedings.mlr.press/v337/du26a.html.

Yann Dubois, Jonathan Gordon, and Andrew YK Foong. Neural process family. http://yanndubs.github.io/Neural-Process-Family/, September 2020.

Marta Garnelo, Dan Rosenbaum, Chris J. Maddison, Tiago Ramalho, David Saxton, Murray Shanahan, Yee Whye Teh, Danilo J. Rezende, and S. M. Ali Eslami. Conditional Neural Processes. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1704–1713. PMLR, 2018a. URL https://proceedings.mlr.press/v80/garnelo18a.html.

Marta Garnelo, Jonathan Schwarz, Dan Rosenbaum, Fabio Viola, Danilo J. Rezende, S. M. Ali Eslami, and Yee Whye Teh. Neural Processes. *arXiv preprint:1807.01622*, 2018b.

Robert E Kass, Valérie Ventura, and Can Cai. Statistical smoothing of neuronal data. *Network: Computation in Neural Systems*, 14(1):5–15, 2003.

Diederik P. Kingma and Max Welling. Auto-Encoding variational Bayes. In *International Conference on Learning Representations*, 2014.

Chris Lloyd, Tom Gunter, Michael A. Osborne, and Stephen J. Roberts. Variational inference for Gaussian process modulated Poisson processes. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 1814–1822. PMLR, 2015. URL https://proceedings.mlr.press/v37/lloyd15.html.

Gabriel Loaiza-Ganem, Sean M. Perkins, Karen E. Schroeder, Mark M. Churchland, and John P. Cunningham. Deep random splines for point process intensity estimation of neural population data. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

George O. Mohler, Martin B. Short, P. Jeffrey Brantingham, Frederic Paik Schoenberg, and George E. Tita. Self-exciting point process modeling of crime. *Journal of the American Statistical Association*, 106(493):100–108, 2011. doi: 10.1198/jasa.2011.ap09546.

Jesper Møller, Anne Randi Syversveen, and Rasmus Plenge Waagepetersen. Log Gaussian Cox processes. *Scandinavian Journal of Statistics*, 25(3):451–482, 1998. doi: 10.1111/1467-9469.00115.

Yosihiko Ogata. Statistical models for Earthquake occurrences and residual analysis for point processes. *Journal of the American Statistical Association*, 83(401):9–27, 1988. doi: 10.1080/01621459.1988.10478560.

Yosihiko Ogata. Space-Time point-process models for Earthquake occurrences. *Annals of the Institute of Statistical Mathematics*, 50(2):379–402, 1998. doi: 10.1023/A:1003403601725.

Takahiro Omi, Naonori Ueda, and Kazuyuki Aihara. Fully neural network based model for general temporal point processes. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Håvard Rue, Sara Martino, and Nicolas Chopin. Approximate Bayesian inference for latent Gaussian models by using integrated nested Laplace approximations. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 71(2):319–392, 2009. doi: 10.1111/j.1467-9868.2008.00700.x.

Déborah Sulem, Vincent Rivoirard, and Judith Rousseau. Bayesian estimation of nonlinear Hawkes processes. *Bernoulli*, 30(2):1257–1286, 2024. doi: 10.3150/23-BEJ1631.

Yoshiaki Takimoto, Yusuke Tanaka, Tomoharu Iwata, Maya Okawa, Hideaki Kim, Hiroyuki Toda, and Takeshi Kurashima. Meta-learning for neural network-based temporal point processes. *arXiv preprint:2401.15846*, 2024.

Shuwan Wang, Christopher K. Wikle, Athanasios C. Micheas, Jessica L. Mark Welch, Jacqueline R. Starr, and Kyu Ha Lee. Inference for stationary Log-Gaussian Cox point processes using Bayesian deep learning: Application to human oral microbiome image data. *Spatial Statistics*, 73:100973, 2026. doi: 10.1016/j.spasta.2026.100973.

Mike Wu, Kristy Choi, Noah Goodman, and Stefano Ermon. Meta-amortized variational inference and learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 6404–6412, 2020. doi: 10.1609/aaai.v34i04.6111.

Qiang Zhang, Aldo Lipani, Omer Kirnap, and Emine Yilmaz. Self-attentive Hawkes process. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 11183–11193. PMLR, 2020. URL https://proceedings.mlr.press/v119/zhang20q.html.

Simiao Zuo, Haoming Jiang, Zichong Li, Tuo Zhao, and Hongyuan Zha. Transformer Hawkes Process. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 11692–11702. PMLR, 2020. URL https://proceedings.mlr.press/v119/zuo20a.html.

A Encoder and Decoder Architectures

A.1 Context Encoder

Let $\mathcal{T}_{\text{train}}$ be the collection of event times in the training tasks. The standardized event time used in Equation 8 is

$$m_t = \frac{1}{|\mathcal{T}_{\text{train}}|} \sum_{t \in \mathcal{T}_{\text{train}}} t, \qquad s_t = \sqrt{\frac{1}{|\mathcal{T}_{\text{train}}|} \sum_{t \in \mathcal{T}_{\text{train}}} (t - m_t)^2}, \qquad \bar{t}_{\tau,k,i} = \frac{t_{\tau,k,i} - m_t}{s_t}. \tag{23}$$

The statistics m_t and s_t are fixed for validation and test tasks.

For compactness, define

$$c_{\tau,k} = \log(1 + N_{\tau,k}), \qquad \bar{\boldsymbol{x}}_\tau = \frac{1}{K_\tau} \sum_{k=1}^{K_\tau} \boldsymbol{x}_{\tau,k},$$

$$\boldsymbol{v}_\tau = \frac{1}{K_\tau} \sum_{k=1}^{K_\tau} (\boldsymbol{x}_{\tau,k} - \bar{\boldsymbol{x}}_\tau)^{\odot 2}, \qquad \kappa_\tau = \log K_\tau.$$

Table 5 presents the encoder as a sequence of modules.

Table 5: Context encoder architecture.

Module	Input	Transformation	Output
Event encoder f_e	$\bar{t}_{\tau,k,i} \in \mathbb{R}$	$\mathrm{MLP}_{\mathrm{SiLU}}(1, 512, 512, 512, 256)$	$\boldsymbol{h}_{\tau,k,i} \in \mathbb{R}^{256}$
Realization representation	$\{\boldsymbol{h}_{\tau,k,i}\}_{i=1}^{N_{\tau,k}}$	$\boldsymbol{u}_{\tau,k} = N_{\tau,k}^{-1} \sum_i \boldsymbol{h}_{\tau,k,i}$; $\boldsymbol{x}_{\tau,k} = [\boldsymbol{u}_{\tau,k}^\top, c_{\tau,k}]^\top$	$\boldsymbol{x}_{\tau,k} \in \mathbb{R}^{257}$
Task statistics	$\{\boldsymbol{x}_{\tau,k}\}_{k=1}^{K_\tau}$	$\boldsymbol{x}_{\mathrm{stat},\tau} = [\bar{\boldsymbol{x}}_\tau^\top, \boldsymbol{v}_\tau^\top, \kappa_\tau]^\top$	$\boldsymbol{x}_{\mathrm{stat},\tau} \in \mathbb{R}^{515}$
Mean head f_μ	$\boldsymbol{x}_{\mathrm{stat},\tau} \in \mathbb{R}^{515}$	$\mathrm{Linear}(515, 256)$	$\boldsymbol{\mu}_{\mathbb{D}_\tau} \in \mathbb{R}^{256}$
Log-variance head f_σ	$\boldsymbol{x}_{\mathrm{stat},\tau} \in \mathbb{R}^{515}$	$\mathrm{MLP}_{\mathrm{SiLU}}(515, 128, 256)$	$\boldsymbol{\ell}_{\mathbb{D}_\tau} \in \mathbb{R}^{256}$

These two heads parameterize the diagonal Gaussian

$$q_\phi(\mathbf{z}_\tau \mid \mathbb{D}_\tau) = \mathcal{N}\big(\boldsymbol{\mu}_{\mathbb{D}_\tau}, \mathrm{diag}(\boldsymbol{\sigma}^2_{\mathbb{D}_\tau})\big), \qquad \boldsymbol{\sigma}^2_{\mathbb{D}_\tau} = \exp(\boldsymbol{\ell}_{\mathbb{D}_\tau}). \tag{24}$$

For numerical stability, the raw output of the log-variance head is mapped element-wise to $(-6, 3)$. We then draw a differentiable sample using the reparameterization trick:

$$\mathbf{z}_\tau = \boldsymbol{\mu}_{\mathbb{D}_\tau} + \exp\big(\tfrac{1}{2}\boldsymbol{\ell}_{\mathbb{D}_\tau}\big) \odot \boldsymbol{\epsilon}, \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}). \tag{25}$$

A.2 Intensity Decoder

The decoder f_λ concatenates a latent sample $\mathbf{z}_\tau \in \mathbb{R}^{256}$ with a query time t. This input passes through three hidden layers of width 512 with SiLU activations, followed by a linear output layer and a softplus transformation that produces the corresponding intensity $\lambda_\theta(t \mid \mathbf{z}_\tau)$.

B Training Objective

This section distinguishes three related ELBOs. The first two belong to a latent-variable model with Gaussian prior $p(\mathbf{z}_\tau) = \mathcal{N}(\mathbf{0}, \boldsymbol{I})$. The third is the objective used to train our learned conditional model.

1. Gaussian-prior ELBO. Combining the Gaussian prior with the likelihood $p_\theta(\mathbb{D}_\tau \mid \mathbf{z}_\tau)$ gives the standard ELBO in Equation 15:

$$\begin{aligned}
\log p_\theta(\mathbb{D}_\tau) &= \log \int p_\theta(\mathbb{D}_\tau \mid \mathbf{z}_\tau) p(\mathbf{z}_\tau)\, \mathrm{d}\mathbf{z}_\tau \\
&= \log \int q_\phi(\mathbf{z}_\tau \mid \mathbb{D}_\tau) \frac{p_\theta(\mathbb{D}_\tau \mid \mathbf{z}_\tau) p(\mathbf{z}_\tau)}{q_\phi(\mathbf{z}_\tau \mid \mathbb{D}_\tau)}\, \mathrm{d}\mathbf{z}_\tau \\
&\geq \mathbb{E}_{q_\phi(\mathbf{z}_\tau \mid \mathbb{D}_\tau)}[\log p_\theta(\mathbb{D}_\tau \mid \mathbf{z}_\tau) + \log p(\mathbf{z}_\tau) - \log q_\phi(\mathbf{z}_\tau \mid \mathbb{D}_\tau)] \\
&= \mathbb{E}_{q_\phi(\mathbf{z}_\tau \mid \mathbb{D}_\tau)}[\log p_\theta(\mathbb{D}_\tau \mid \mathbf{z}_\tau)] - D_{\mathrm{KL}}(q_\phi(\mathbf{z}_\tau \mid \mathbb{D}_\tau) \,\|\, p(\mathbf{z}_\tau)) =: \mathcal{E}_\tau.
\end{aligned} \tag{26}$$

2. Gaussian-prior conditional ELBO. During training, we partition the realizations into disjoint sets $\mathbb{D}_\tau = \mathbb{C}_\tau \cup \mathbb{T}_\tau$, with $|\mathbb{C}_\tau| = K_C$, $|\mathbb{T}_\tau| = K_T$, and $K_\tau = K_C + K_T$. Under the same Gaussian-prior model, the context induces $p_\theta(\mathbf{z}_\tau \mid \mathbb{C}_\tau)$, and the conditional evidence is

$$p_\theta(\mathbb{T}_\tau \mid \mathbb{C}_\tau) = \int p_\theta(\mathbb{T}_\tau \mid \mathbf{z}_\tau) p_\theta(\mathbf{z}_\tau \mid \mathbb{C}_\tau)\, \mathrm{d}\mathbf{z}_\tau. \tag{27}$$

Using $q_\phi(\mathbf{z}_\tau \mid \mathbb{C}_\tau \cup \mathbb{T}_\tau)$ gives

$$\log p_\theta(\mathbb{T}_\tau \mid \mathbb{C}_\tau) \geq \mathbb{E}_{q_\phi(\mathbf{z}_\tau \mid \mathbb{C}_\tau \cup \mathbb{T}_\tau)}[\log p_\theta(\mathbb{T}_\tau \mid \mathbf{z}_\tau)] - D_{\mathrm{KL}}(q_\phi(\mathbf{z}_\tau \mid \mathbb{C}_\tau \cup \mathbb{T}_\tau) \,\|\, p_\theta(\mathbf{z}_\tau \mid \mathbb{C}_\tau)) =: \mathcal{E}_\tau^{\mathrm{cond}}. \tag{28}$$

3. Learned conditional ELBO (our objective). Our model does not use the Gaussian-prior posterior $p_\theta(\mathbf{z}_\tau \mid \mathbb{C}_\tau)$ as its conditional prior. Instead, it uses the learned encoder distribution $q_\phi(\mathbf{z}_\tau \mid \mathbb{C}_\tau)$, defining

$$\widetilde{p}_{\phi,\theta}(\mathbb{T}_\tau \mid \mathbb{C}_\tau) = \int p_\theta(\mathbb{T}_\tau \mid \mathbf{z}_\tau) q_\phi(\mathbf{z}_\tau \mid \mathbb{C}_\tau)\, \mathrm{d}\mathbf{z}_\tau. \tag{29}$$

Applying Jensen's inequality with $q_\phi(\mathbf{z}_\tau \mid \mathbb{C}_\tau \cup \mathbb{T}_\tau)$ gives the objective used in Equation 19:

$$\log \widetilde{p}_{\phi,\theta}(\mathbb{T}_\tau \mid \mathbb{C}_\tau) \geq \mathbb{E}_{q_\phi(\mathbf{z}_\tau \mid \mathbb{C}_\tau \cup \mathbb{T}_\tau)}[\log p_\theta(\mathbb{T}_\tau \mid \mathbf{z}_\tau)] - D_{\mathrm{KL}}(q_\phi(\mathbf{z}_\tau \mid \mathbb{C}_\tau \cup \mathbb{T}_\tau) \,\|\, q_\phi(\mathbf{z}_\tau \mid \mathbb{C}_\tau)) =: \mathcal{E}_\tau^{\mathrm{amortized}}. \tag{30}$$

Thus, $\mathcal{E}_\tau^{\mathrm{amortized}}$ is a valid ELBO for the learned conditional evidence. If $q_\phi(\mathbf{z}_\tau \mid \mathbb{C}_\tau) = p_\theta(\mathbf{z}_\tau \mid \mathbb{C}_\tau)$, the second and third ELBOs coincide.

C Synthetic Experimental Details

C.1 Data Generation and Dataset Splits

We generate 5,500 tasks on $[0, T]$ with $T = 10$. Each task has one latent intensity and 16 iid realizations. As in Section 4.1.1,

$$f_\tau \sim \mathrm{GP}(0, C_{\mathrm{RBF}}), \qquad \lambda_\tau(t) = 20\,\mathrm{softplus}(f_\tau(t)), \tag{31}$$

where $C_{\mathrm{RBF}}(t, t') = \sigma_f^2 \exp(-(t-t')^2/(2\ell^2))$, with signal variance $\sigma_f^2 = 1$ and lengthscale $\ell = 2$. We sample each GP on an equally spaced grid of $G = 300$. The same grid is used for intensity evaluation.

The synthetic dataset contains 4,500 training, 500 validation, and 500 test tasks. Splits are disjoint at the task level. Table 6 summarizes event counts using all 16 realizations of every task. Event time standardization is estimated from the training split and held fixed for validation, testing, and OOD evaluation.

Table 6: Synthetic dataset statistics.

Split	Tasks	Sequences	Events	Events/sequence	Range
Train	4,500	72,000	11,625,135	161.46 ± 67.80	15–484
Validation	500	8,000	1,265,973	158.25 ± 68.72	7–441
Test	500	8,000	1,275,955	159.49 ± 64.91	13–444

C.2 Training and Checkpoint Selection

Each update samples a batch of 256 training tasks. We draw K_C once per batch from the mixture specified in Section 4.1.1 and set $K_T = K_{\mathrm{train}} - K_C$. Each task's realizations are partitioned into K_C context and K_T target realizations. We train each $K_{\mathrm{train}} \in \{4, 8, 16\}$ configuration for 20,000 updates with seeds 42–45, using AdamW with initial learning rate 10^{-3}, weight decay 10^{-4}, and cosine learning rate decay.

We save a checkpoint every 1,000 updates. After training, all 20 checkpoints are evaluated on the 500 validation tasks. We hold out each realization in turn and use the remaining realizations as context. The checkpoint with the highest validation HLL is selected separately for each training seed. Neither ground-truth intensity labels nor test tasks are used for selection.

Table 7 reports the validation HLL of every saved checkpoint for seed 42. For all three values of K_{train}, the checkpoint at 20,000 updates achieves the highest validation HLL and is therefore selected.

Table 7: Validation HLL across saved checkpoints for seed 42. Values are shown to two decimal places.

	Validation HLL ↑		
Update	$K_{\text{train}} = 4$	$K_{\text{train}} = 8$	$K_{\text{train}} = 16$
1,000	308.32	308.69	309.38
2,000	310.11	309.21	310.71
3,000	310.76	309.61	310.85
4,000	310.62	309.89	310.90
5,000	310.81	310.11	311.02
6,000	310.84	310.28	311.04
7,000	310.90	310.28	310.85
8,000	310.94	310.26	311.08
9,000	310.98	310.31	311.00
10,000	310.99	310.36	311.18
11,000	310.91	310.38	311.14
12,000	310.96	310.37	311.19
13,000	311.00	310.41	311.19
14,000	311.00	310.40	311.20
15,000	311.01	310.40	311.20
16,000	311.01	310.41	311.21
17,000	311.02	310.41	311.22
18,000	311.02	310.42	311.22
19,000	311.03	310.43	311.22
20,000	**311.03**	**310.43**	**311.23**

To estimate HLL in Equation 22, we draw $\mathbf{z}_\tau^{(j)} \sim q_\phi(\mathbf{z}_\tau \mid \mathbb{D}_{\tau,-h})$ for each held-out realization h. Its point process log likelihood under sample j is

$$\ell_{\tau,h}^{(j)} = \sum_{t_i \in \mathbb{S}_{\tau,h}} \log \lambda_\theta(t_i \mid \mathbf{z}_\tau^{(j)}) - \int_0^T \lambda_\theta(t \mid \mathbf{z}_\tau^{(j)})\, \mathrm{d}t. \tag{32}$$

For $s \in \{\text{val}, \text{test}\}$, the Monte Carlo estimate is

$$\widehat{\text{HLL}}_s = \frac{1}{|\mathcal{D}_s|} \sum_{\mathbb{D}_\tau \in \mathcal{D}_s} \frac{1}{|\mathbb{D}_\tau|} \sum_{h=1}^{|\mathbb{D}_\tau|} \log \left[\frac{1}{n_{\text{MC}}} \sum_{j=1}^{n_{\text{MC}}} \exp\{\ell_{\tau,h}^{(j)}\} \right]. \tag{33}$$

We compute the inner log average using log-mean-exp with $n_{\text{MC}} = 300$.

C.3 Evaluation Protocol and Metrics

To evaluate generalization to numbers of realizations not seen during training, we retain the 16 realizations of each of the 500 test tasks and generate 48 additional realizations from the same underlying intensity. Evaluation uses the first $K_{\text{test}} \in \{1, 2, 4, 8, 16, 32, 64\}$ realizations, giving nested contexts as replication increases. The underlying test intensities and contexts are shared across methods.

For $n_{\text{test}} = 500$ test tasks, let $\lambda_{\tau,g}$ be the ground-truth intensity at grid point t_g, $g = 1, \ldots, G$. Let $\mathbf{z}_\tau^{(m)}$, $m = 1, \ldots, n_{\text{MC}}$, denote $n_{\text{MC}} = 300$ draws from $q_\phi(\mathbf{z}_\tau \mid \mathbb{D}_\tau)$, and set $\lambda_{\tau,g}^{(m)} = \lambda_\theta(t_g \mid \mathbf{z}_\tau^{(m)})$. The posterior mean intensity is

$$\widehat{\lambda}_{\tau,g} = \frac{1}{n_{\text{MC}}} \sum_{m=1}^{n_{\text{MC}}} \lambda_{\tau,g}^{(m)}.$$

Let $\ell_{\tau,g}$ and $u_{\tau,g}$ be the pointwise 2.5% and 97.5% sample quantiles. The task-averaged metrics are

$$\begin{aligned}
\mathrm{RMSE} &= \frac{1}{n_{\mathrm{test}}} \sum_{\tau=1}^{n_{\mathrm{test}}} \sqrt{\frac{1}{G} \sum_{g=1}^{G} (\widehat{\lambda}_{\tau,g} - \lambda_{\tau,g})^2}, \\
\mathrm{C95} &= \frac{1}{n_{\mathrm{test}}} \sum_{\tau=1}^{n_{\mathrm{test}}} \left[\frac{1}{G} \sum_{g=1}^{G} \mathbf{1}\{\ell_{\tau,g} \le \lambda_{\tau,g} \le u_{\tau,g}\} \right], \\
\mathrm{IW95} &= \frac{1}{n_{\mathrm{test}}} \sum_{\tau=1}^{n_{\mathrm{test}}} \left[\frac{1}{G} \sum_{g=1}^{G} (u_{\tau,g} - \ell_{\tau,g}) \right].
\end{aligned} \tag{34}$$

C.4 Additional Training-Replication Results

Table 8 reports the corresponding results for models trained with $K_{\mathrm{train}} \in \{8, 16\}$. Both configurations show lower RMSE and narrower CIs as K_{test} increases, while empirical coverage remains broadly stable. Compared with the $K_{\mathrm{train}} = 4$ model in Table 3, these models achieve lower RMSE when many test realizations are available, but slightly higher RMSE when K_{test} is small.

Table 8: Evidence accumulation for models trained with $K_{\mathrm{train}} \in \{8, 16\}$. Results are averaged over four seeds.

	RMSE ↓		**C95 (%)**		**IW95**	
K_{test}	$K_{\mathrm{train}} = 8$	$K_{\mathrm{train}} = 16$	$K_{\mathrm{train}} = 8$	$K_{\mathrm{train}} = 16$	$K_{\mathrm{train}} = 8$	$K_{\mathrm{train}} = 16$
1	$2.75_{\pm 0.01}$	$2.81_{\pm 0.01}$	$92.23_{\pm 0.16}$	$89.81_{\pm 2.62}$	$9.60_{\pm 0.03}$	$9.14_{\pm 0.61}$
2	$2.03_{\pm 0.01}$	$2.06_{\pm 0.01}$	$92.66_{\pm 0.10}$	$90.64_{\pm 2.35}$	$7.12_{\pm 0.01}$	$6.77_{\pm 0.44}$
4	$1.56_{\pm 0.01}$	$1.56_{\pm 0.01}$	$91.90_{\pm 0.16}$	$89.73_{\pm 1.77}$	$5.37_{\pm 0.01}$	$5.04_{\pm 0.24}$
8	$1.19_{\pm 0.01}$	$1.18_{\pm 0.01}$	$92.78_{\pm 0.26}$	$90.94_{\pm 0.57}$	$4.23_{\pm 0.01}$	$3.89_{\pm 0.06}$
16	$0.93_{\pm 0.01}$	$0.91_{\pm 0.01}$	$93.74_{\pm 0.31}$	$92.40_{\pm 0.74}$	$3.54_{\pm 0.01}$	$3.18_{\pm 0.07}$
32	$0.79_{\pm 0.01}$	$0.75_{\pm 0.01}$	$93.46_{\pm 0.20}$	$92.95_{\pm 1.33}$	$3.12_{\pm 0.01}$	$2.76_{\pm 0.13}$
64	$0.71_{\pm 0.02}$	$0.66_{\pm 0.02}$	$93.12_{\pm 0.22}$	$93.19_{\pm 1.79}$	$2.89_{\pm 0.01}$	$2.53_{\pm 0.17}$

C.5 Baseline Implementations

PSTH. We divide pooled bin counts by the number of input realizations and the bin width. We select the bin width on the 500 validation tasks using per-task intensity RMSE from four input realizations. The selected width 1.0 is fixed for test evaluation.

KDE. We use a boundary-normalized Gaussian kernel and select its bandwidth using the same validation RMSE. The selected bandwidth 0.5 is fixed for four-realization test evaluation. Table 9 gives both validation searches.

LGCP. The LGCP baseline is fitted directly to each test task using an RBF prior, an exponential link, 10 inducing points, and 10,000 variational optimization steps per test task. We use a Gaussian variational distribution over the inducing variables and jointly optimize the GP mean, lengthscale, and signal variance. Metrics use 300 intensity samples.

FIM-PP. We use the released `FIM4Science/fim-pp` checkpoint without fine-tuning. Its context contains the first four realizations, and its inference sequence is the fifth realization. It therefore receives an additional observed sequence compared with the other baselines.

Table 10 summarizes the compute environment and training or fitting budget for Ours, LGCP, and FIM-PP.

C.6 Out-of-Distribution Test

Both OOD evaluations use the $K_{\mathrm{train}} = 8$ models and $K_{\mathrm{test}} = 8$. Each test set contains 500 tasks.

Table 9: Validation selection for deterministic baselines on the synthetic dataset.

PSTH bin width	Validation RMSE	KDE bandwidth	Validation RMSE
0.05	8.67	0.05	4.65
0.10	6.13	0.10	3.32
0.20	4.33	0.20	2.42
0.30	4.93	0.30	2.08
0.50	2.80	0.50	**1.87**
0.75	5.78	0.75	2.03
1.00	**2.40**	1.00	2.40
Selected bin width	1.00	Selected bandwidth	0.50
Test RMSE	2.45	Test RMSE	1.91

Table 10: Synthetic compute and implementation record.

Method	Hardware	Training time	Checkpoint / fit budget
Ours, $K_{\text{train}} = 4$	NVIDIA GH200	2,969 s	20,000 updates
Ours, $K_{\text{train}} = 8$	NVIDIA GH200	3,497 s	20,000 updates
Ours, $K_{\text{train}} = 16$	NVIDIA GH200	4,305 s	20,000 updates
LGCP	NVIDIA GH200	—	10 inducing points; 10,000 steps per task
FIM-PP	NVIDIA GH200	Pretrained	Released checkpoint

OOD1: constant intensity. We draw a constant intensity uniformly from $[5, 20]$ and generate 8 realizations.

OOD2: GP–constant–linear intensity. We draw transition times $b_1 \sim \text{Unif}(2, 4)$ and $b_2 \sim \text{Unif}(6, 8)$ and set c to the original GP intensity at b_1. The transformed intensity is

$$\lambda_\tau^{\text{OOD2}}(t) = \begin{cases} \lambda_\tau(t), & 0 \le t < b_1, \\ c, & b_1 \le t < b_2, \\ c[1 + r(t - b_2)], & b_2 \le t \le T, \end{cases} \tag{35}$$

where $r \sim \text{Unif}(-0.9/(T - b_2),\ 1.5/(T - b_2))$. Thus the final intensity is between $0.1c$ and $2.5c$.

Across four training seeds, OOD1 attains C95 of 95.72, while OOD2 has lower coverage of 91.99 and higher RMSE (Table 11). Representative tasks are shown in Figure 4.

Table 11: OOD results. Results are averaged over four training seeds.

Test set	RMSE	C95 (%)	IW95
OOD1	$0.89_{\pm 0.01}$	$95.72_{\pm 0.10}$	$3.77_{\pm 0.01}$
OOD2	$1.23_{\pm 0.02}$	$91.99_{\pm 0.17}$	$4.25_{\pm 0.01}$

D Real-Data Protocol and Metrics

D.1 Data Construction

We use the Divvy bicycle-share trip records from the City of Chicago Open Data Portal for 2013–2019. Records include origin and destination stations and trip timestamps; we use stations and departure times. It contains 21,242,740 trips. We treat each trip start as an event and group starts by origin station and day, producing 951,508 station-day sequences.

A task is a station–year–month–weekday group with at least four observed days. We retain the first four days chronologically as its $K = 4$ realizations, yielding 194,334 candidate tasks. We first compare the four total event counts under an equal rate Poisson null using a Pearson chi-square test. We retain 148,151 tasks.

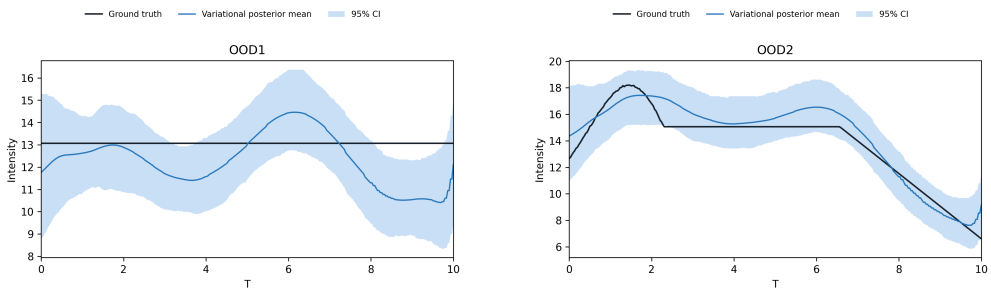


Figure 4: OOD intensity recovery at $K_{\text{test}} = 8$ for OOD1 (left) and OOD2 (right).

We divide each day into 10 equal-width time bins and use a Pearson test to assess whether the four days share the same normalized temporal profile. Tasks with $p \geq 0.01$ are retained.

We select the top 10% by total events across the four realizations. This yields 13,468 tasks and a cutoff of 163 events per task. Stations are then assigned disjointly to training, validation, and test splits, giving 10,774/1,347/1,347 tasks. Finally, we sample half of the tasks within each split and obtain the final dataset. Table 12 summarizes the resulting dataset.

Table 12: Divvy dataset after preprocessing.

Split	Tasks	Sequences	Stations	Events	Events/task
Train	5,387	21,548	182	1,437,806	163–1,534
Validation	674	2,696	21	170,078	163–1,018
Test	674	2,696	21	184,032	163–1,154

D.2 Training and Checkpoint Selection

Our model is trained on the 5,387 tasks in $\mathcal{D}_{\text{train}}$. Feature normalization statistics are computed from the training split only. We use AdamW with initial learning rate 10^{-3}, cosine decay, weight decay 10^{-4}, and batches of 256 tasks for 20,000 updates. For each task, the context size K_C is sampled from $\frac{1}{2}\,\text{Uniform}\{2,3\} + \frac{1}{2}\,\text{Uniform}\{1,2,3\}$. We save a checkpoint every 1,000 updates. After training, all 20 checkpoints are evaluated on the 674 tasks in $\mathcal{D}_{\text{val}}$ using the four-fold HLL evaluation in Appendix C.2, with $n_{\text{MC}} = 300$ latent samples. The checkpoint with the highest validation HLL is selected for each seed. Table 13 reports the validation HLL of all saved checkpoints for seeds 42–45.

D.3 Baseline Selection and Implementation

We select the PSTH bin width and KDE bandwidth by four-fold leave-one-realization-out likelihood on all 674 validation tasks. Both searches use the candidate grid $\{0.05, 0.1, 0.2, 0.3, 0.5, 0.75, 1.0\}$. The selected values are 1.0 for PSTH and 0.3 for KDE. For both methods, HLL is computed using their point estimates of the intensity.

LGCP is fit independently in each fold using 10 inducing points and 10,000 optimization steps. Its HLL is computed by averaging the point process likelihood over 300 samples from its variational intensity distribution and then taking the logarithm, following the same Monte Carlo evaluation as our method.

FIM-PP requires one inference sequence in addition to its context. For each held-out realization, we therefore run FIM-PP three times. Each visible realization serves once as the inference sequence, while the other two form the context. We score the held-out realization under each resulting intensity, average the three likelihoods, and then take the logarithm.

Table 13: Validation HLL across saved checkpoints on Divvy. Bold indicates the best checkpoint for each seed, determined before rounding.

	Validation HLL ↑			
Update	Seed 42	Seed 43	Seed 44	Seed 45
1,000	82.87	82.63	81.67	81.95
2,000	83.62	83.61	83.30	82.95
3,000	83.55	83.75	83.72	83.59
4,000	83.45	83.71	83.83	83.47
5,000	83.99	84.00	83.87	83.99
6,000	83.84	83.87	84.03	83.97
7,000	84.14	84.10	83.97	84.23
8,000	84.19	84.18	83.97	84.16
9,000	84.25	84.16	84.19	83.77
10,000	84.21	84.27	84.26	84.22
11,000	84.23	84.23	84.19	84.23
12,000	84.14	84.25	84.22	84.25
13,000	84.28	84.30	84.25	84.26
14,000	84.27	84.32	84.31	84.27
15,000	84.30	84.29	84.30	84.29
16,000	84.29	84.31	84.30	84.30
17,000	84.31	**84.34**	84.31	84.31
18,000	**84.32**	84.34	**84.33**	**84.32**
19,000	84.32	84.33	84.32	84.31
20,000	84.32	84.34	84.33	84.31