# SHARING TEMPORAL STRUCTURE FOR JOINT TIME–MARK PREDICTION

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Distinct event types can exhibit different timing profiles without requiring separate temporal components. We introduce Shared Temporal Mixture (STMix), a history-conditioned joint time–mark model that uses shared temporal structure for both learning and probability queries. Each mixture component pairs a temporal density with a categorical mark distribution, allowing each type to combine the same temporal components with different weights. During joint likelihood training, observed times and marks determine component responsibilities, enabling events of different types to shape common temporal patterns. Tractable component cumulative distributions express marked next-event window probabilities and forecasts after an event-free wait as finite sums without numerical time integration. A query algorithm prepares components once per observed history and reuses endpoint evaluations across types, horizons, and requests. Experiments on four real-world datasets demonstrate competitive joint prediction; matched structural comparisons on three datasets achieve better test likelihood with 69–86% fewer total parameters than type-specific models. Query experiments show competitive window forecasts, while endpoint reuse yields 1.38–2.60$\times$ speedups over a vectorized implementation of the same model without caching.

## 1 INTRODUCTION

Event prediction often calls for probabilities, not just point estimates. In online activity modeling, for example, we may ask how likely the next action is to have a given type within a specified window. After a period without activity, the forecast must incorporate that additional observation. Joint time–mark distributions support such queries (Boyd et al., 2023) and uncertainty estimates that respect time–mark dependence (Dheur et al., 2024). The challenge is to learn this distribution accurately and make its probabilities efficient to compute.

Neural point processes learn temporal dependencies through recurrent networks, self-attention, and continuous-time state-space models (Du et al., 2016; Mei & Eisner, 2017; Zuo et al., 2020; Chang et al., 2025). What a model can learn and compute also depends on its output distribution (Shchur et al., 2021). Learned cumulative intensities and flexible event densities connect expressive modeling with tractable likelihood evaluation (Omi et al., 2019; Shchur et al., 2020a; Draxler et al., 2025). Unified empirical comparisons further show that decoder design affects both time and mark prediction (Bosser & Ben Taieb, 2023). We focus on a specific design choice: how temporal structure is organized across event types.

Existing density models make different choices about this structure. The original marked extension of LogNormMix predicts time and type independently given the history (Shchur et al., 2020a). Assigning a separate temporal mixture to each type captures time–mark dependence while retaining tractable cumulative probabilities (Waghmare et al., 2022). However, the temporal component bank then grows with the number of types. Sharing the history encoder still leaves the output densities separate. A common component bank would instead let different event types contribute to learning the same temporal patterns.

Probability queries introduce a second requirement. A time-dependent mark classifier can capture time–mark dependence without a separate time density for each type (Bosser & Ben Taieb, 2025). Yet a tractable time marginal is not enough. Marked window probabilities generally require inte-



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.

grating the product of the time density and the classifier output. Can shared temporal components instead support both distinct type-specific timing profiles and efficient probability queries?

**Distinct timing profiles need not require distinct temporal components.** We introduce *Shared Temporal Mixture* (STMix), a history-conditioned mixture whose components pair temporal densities with categorical mark distributions. Each type combines the same temporal components with different weights. This captures time–mark dependence while keeping the temporal head compact. During joint likelihood training, both the observed time and type determine component responsibility. Events of different types can therefore shape common temporal patterns. We instantiate the model with a recurrent history encoder and four log-time rational-quadratic spline components.

The shared components also make probability computation reusable. With tractable component CDFs, mark marginals, marked next-event window probabilities, and survival-conditioned forecasts reduce to finite sums. Our query algorithm prepares component parameters once per observed history and evaluates each unique endpoint only once. It reuses these results across types, horizons, and requests, without numerical time integration. Survival-based conditioning updates component weights and remaining-time probabilities during an event-free wait, without re-encoding the history. The same temporal components are thus learned across types and reused across queries.

We evaluate STMix on four real-world datasets. It achieves lower mean test joint negative log-likelihood than the LNM-Joint64 + summary, FlexTPP, and AttNHP baselines on all four. Matched structural comparisons contrast shared and type-specific temporal heads on three datasets, using the same history encoder, temporal density family, and training protocol. At validation-selected checkpoints, sharing improves test likelihood in all nine dataset–seed pairs, with 69–86% fewer total parameters.

Query experiments evaluate both forecast quality and computational cost. STMix achieves lower mean test window log loss and Brier score than LNM-Joint64 + summary on all three evaluated datasets. In a GPU benchmark on MOOC, endpoint reuse yields a 1.38–2.60$\times$ speedup on mixed probability-query workloads over a vectorized implementation of the same fixed model without endpoint caching. Together, the prediction and query results show how temporal sharing supports compact joint modeling and reusable probabilistic inference.

Our contributions are threefold:

- **Joint modeling through shared temporal structure.** We introduce STMix, a compact, history-conditioned joint density that lets event types jointly learn temporal components while retaining distinct timing profiles.
- **A reusable probability-query algorithm.** We develop a component-level query algorithm that prepares a history-specific query state and reuses endpoint probabilities across event types, horizons, and requests. It computes mark marginals, marked next-event window probabilities, and survival-conditioned forecasts through finite sums without numerical time integration.
- **Predictive and computational benefits of temporal sharing.** Matched structural comparisons demonstrate better test likelihood with fewer parameters. Query benchmarks establish window-prediction quality and isolate the computational gains from reuse.

## 2 Related Work

**Joint Time–Mark Distributions** Statistical dependence and temporal parameter sharing are distinct choices. The original marked LogNormMix extension predicts time and mark independently given the history (Shchur et al., 2020a). Mark-conditioned mixtures introduce dependence through type-specific temporal components (Waghmare et al., 2022). Time-conditioned mark predictors instead couple a common time marginal to a conditional mark distribution (Bosser & Ben Taieb, 2025). DTPP combines this factorization with log-normal gap mixtures conditioned on the previous mark and a Transformer mark predictor (Panos, 2024). STMix shares temporal components across type-conditional time densities, with type-dependent weights retaining distinct timing profiles.

**Temporal Sharing and Joint Learning** Product mixtures of continuous kernels and categorical distributions appear in Bayesian marked Poisson-process models (Taddy & Kottas, 2012). We use

this product-mixture structure in a history-conditioned next-event model, where observed times and marks jointly determine the responsibilities used to train shared temporal components. Bosser & Ben Taieb (2025) analyze conflicting gradients and develop disjoint decoder and history-encoder parameterizations while preserving time–mark dependence. Our posterior-stop control tests direct mark-loss feedback through density-posterior weights while retaining the shared encoder.

**Probabilistic Querying** Learned cumulative intensities (Omi et al., 2019) and density mixtures (Shchur et al., 2020a) support likelihood evaluation without numerical integration. Dec-ODE separates the evolving influences of past events and jointly evaluates densities and survival probabilities through an ODE solve (Song et al., 2024). Mark-specific log-normal mixtures yield analytic marked next-event window probabilities (Waghmare et al., 2022), while IFNMTPP directly parameterizes mark-specific joint tails to obtain mark marginals and conditional time CDFs (Liu et al., 2025). STMix assembles marked probabilities from common temporal CDF and survival values, reusing component endpoint evaluations across all types. For sequence-level inference, TriTPP provides parallel likelihood computation and sampling via triangular maps (Shchur et al., 2020b), while Boyd et al. (2023) develop importance-sampling estimators for hitting times, future event types, and event ordering. Our algorithm specializes to immediate next-event queries, including post-wait forecasts, and reuses decoded components and endpoint probabilities across requests at a fixed history.

## 3 METHOD

STMix defines a joint next-event distribution from a shared bank of temporal components (Figure 1). Different event types combine these components with different weights, and joint likelihood training lets their observations shape the same temporal bank. The resulting representation also supports probability queries by reusing component calculations.

### 3.1 A JOINT DISTRIBUTION WITH SHARED TEMPORAL COMPONENTS

Let $\mathcal{H}_i = ((t_1, k_1), \ldots, (t_i, k_i))$ be the observed history, with marks $k_j \in \{1, \ldots, K\}$. We predict the next gap $T = t_{i+1} - t_i$ and mark $Y = k_{i+1}$ from an encoding $h = \mathrm{Enc}(\mathcal{H}_i)$. A scale $s > 0$, fitted on training data, defines normalized time $U = T/s$. We write $\tau$ and $u = \tau/s$ for realizations of $T$ and $U$, respectively. We first describe the positive-gap distribution $p_+(u, k \mid h)$, conditional on $T > 0$; recorded zero gaps are incorporated below.

For each history, the decoder produces mixture weights $\pi \in \Delta^{R-1}$, positive-time densities $\{f_r\}_{r=1}^R$, and categorical distributions $q_r \in \Delta^{K-1}$, where $\Delta^{m-1}$ denotes the probability simplex. Their joint distribution is

$$p_+(u, k \mid h) = \sum_{r=1}^{R} \pi_r(h) f_r(u \mid h) q_r(k \mid h), \qquad u > 0. \tag{1}$$

Each component pairs a temporal pattern with a distribution over event types. Summing over marks gives the time marginal $f_+(u \mid h) = \sum_r \pi_r f_r(u \mid h)$, while integrating over time gives the positive-gap mark marginal $m_k(h) = \sum_r \pi_r q_r(k \mid h)$. In particular,

$$p_+(u \mid Y = k, h) = \sum_{r=1}^{R} \alpha_{rk}(h) f_r(u \mid h), \qquad \alpha_{rk}(h) = \frac{\pi_r(h) q_r(k \mid h)}{m_k(h)}. \tag{2}$$

Thus, each type has its own timing profile through its weights $\alpha_{rk}$, while all types use the same temporal components. Conversely, observing an event at time $u$ changes the predicted type through the density posterior

$$p_+(k \mid u, h) = \sum_{r=1}^{R} w_r(u, h) q_r(k \mid h), \qquad w_r(u, h) = \frac{\pi_r(h) f_r(u \mid h)}{f_+(u \mid h)}. \tag{3}$$

The components and both sets of weights depend on the history. Sharing retains time–mark dependence with a common temporal bank. For a $D$-dimensional history encoding and $d$ parameters per component, the shared linear temporal head uses $Rd(D+1)$ parameters, compared with $KRd(D+1)$ for $R$ separate components per type.

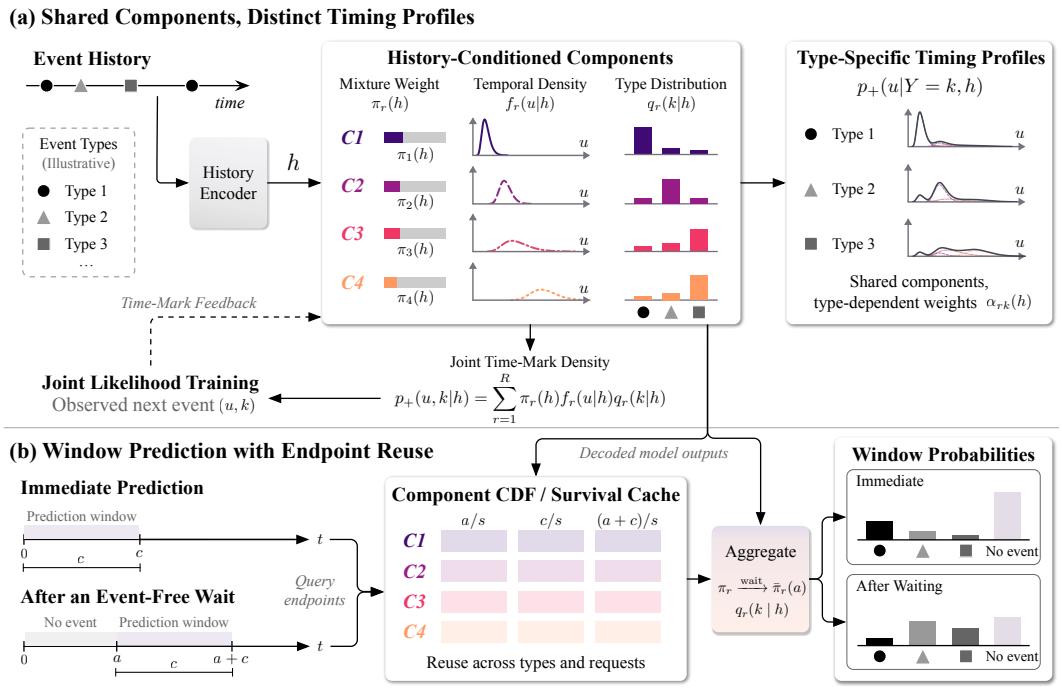



Figure 1: STMix shares temporal structure across event types and probability queries. (a) History-conditioned components form a joint time–mark density and distinct type-specific timing profiles; joint likelihood training shapes the shared components. (b) Cached component CDF and survival values are reused across types and requests to form immediate and post-wait window probabilities, including no-event probabilities. The positive-gap branch is shown.

**Temporal Parameterization** We choose components whose density and cumulative probabilities come from the same transformation, connecting likelihood learning to probability queries. We use $R = 4$ conditional log-time rational-quadratic spline densities (Durkan et al., 2019). For component $r$, let $x_r = (\log u - \mu_r(h))/\sigma_r(h)$ and $z_r = g_r(x_r; h)$, where $\sigma_r > 0$ and $g_r$ is an increasing spline with identity tails. With standard-normal density $\phi$ and CDF $\Phi$, the component density, CDF, and survival function are

$$f_r(u \mid h) = \phi(z_r)\frac{g_r'(x_r;h)}{\sigma_r(h)u}, \qquad F_r(u \mid h) = \Phi(z_r), \qquad S_r(u \mid h) = \Phi(-z_r). \tag{4}$$

A GRU encodes observed marks and $\log(1+u)$ gaps; linear heads predict the temporal parameters and softmax-normalized $\pi$ and $q_r$. Architecture and initialization details appear in Appendix A.1.

### 3.2 Joint Learning Across Event Types

We learn the shared components by joint maximum likelihood. For a positive target $(u, k)$, the loss separates into a time term and a conditional-mark term:

$$\mathcal{L}_{\text{joint}} = -\log p_+(u, k \mid h) = \mathcal{L}_{\text{time}} + \mathcal{L}_{\text{mark}}, \tag{5}$$

where $\mathcal{L}_{\text{time}} = -\log f_+(u \mid h)$ and $\mathcal{L}_{\text{mark}} = -\log p_+(k \mid u, h)$. We evaluate the mixture with log-sum-exp and train both terms together.

**How Marks Shape Temporal Components** The observed time and type jointly determine the responsibility of component $r$:

$$\gamma_r(u, k, h) = \frac{\pi_r(h) f_r(u \mid h) q_r(k \mid h)}{p_+(u, k \mid h)} = \frac{w_r(u, h) q_r(k \mid h)}{\sum_j w_j(u, h) q_j(k \mid h)}. \tag{6}$$

At the component log-scores $\xi_r = \log \pi_r + \log f_r(u \mid h)$, the local derivatives are

$$\frac{\partial \mathcal{L}_{\text{time}}}{\partial \xi_r} = -w_r, \qquad \frac{\partial \mathcal{L}_{\text{mark}}}{\partial \xi_r} = w_r - \gamma_r, \qquad \frac{\partial \mathcal{L}_{\text{joint}}}{\partial \xi_r} = -\gamma_r. \tag{7}$$

The mark term adjusts the time-only responsibility $w_r$ to the joint responsibility $\gamma_r$. Even when several components explain an event's time, its type can favor one component and alter how that component learns. Events of different types thus contribute to common temporal densities through their joint assignments.

To test this feedback path, our experiments include a *posterior-stop* control that detaches $w$ only in $\mathcal{L}_{\text{mark}}$. It preserves the predictive family and shared encoder while blocking direct mark-loss gradients through $w$ to the temporal and mixture heads.

**Complete Observation Model** We retain recorded ties using a zero-gap mass $p_0$, fitted on training data along with $s$. The full next-event law is

$$\begin{aligned} \Pr(T = 0, Y = k \mid h) &= p_0 q^0(k \mid h), \\ p(\tau, k \mid h) &= \frac{1 - p_0}{s} p_+(\tau/s, k \mid h), \qquad \tau > 0, \end{aligned} \tag{8}$$

where $q^0 = q_1$ uses the first categorical component at a fixed index. Training minimizes the negative log-likelihood of this law over all valid next-event targets. The first event only supplies context, and each target is excluded from its own history. Appendices A.2 and A.3 give the complete objective, the training-data estimates of $s$ and $p_0$, and the feedback derivation.

### 3.3 Probability queries through component reuse

Fix a history $h$ and suppress it in the notation. All queries below concern the *immediate next event*. The components support both predictions made immediately after the history and forecasts updated during a period without events.

**Immediate Predictions** For a source-time horizon $b \geq 0$, the full mark marginal, marked window probabilities, and no-event probability are

$$\begin{aligned} \Pr(Y = k) &= p_0 q^0(k) + (1 - p_0) \sum_r \pi_r q_r(k), \\ v_k(b) := \Pr(T \leq b, Y = k) &= p_0 q^0(k) + (1 - p_0) \sum_r \pi_r F_r(b/s) q_r(k), \\ v_\varnothing(b) := \Pr(T > b) &= (1 - p_0) \sum_r \pi_r S_r(b/s). \end{aligned} \tag{9}$$

The $K$ marked-window probabilities and the no-event probability sum to one. The zero-gap mass is included even at $b = 0$.

**Forecasts After an Event-Free Wait** Suppose no event has occurred by a finite time $a \geq 0$, so that $T > a$. Conditioning on this observation changes the component weights to

$$\bar{\pi}_r(a) = \frac{\pi_r S_r(a/s)}{\sum_j \pi_j S_j(a/s)}, \tag{10}$$

where the weights $w_r$, $\gamma_r$, and $\bar{\pi}_r(a)$ are posteriors over the same component bank. They condition on an event time, a time–mark pair, and an event-free wait, respectively.

The next-mark distribution is $\Pr(Y = k \mid T > a) = \sum_r \bar{\pi}_r(a) q_r(k)$. For $b \geq a$, define the within-component residual survival $\rho_r(a, b) = S_r(b/s)/S_r(a/s)$. Then

$$\begin{aligned} \bar{v}_k(a, b) := \Pr(a < T \leq b, Y = k \mid T > a) &= \sum_r \bar{\pi}_r(a) \big[1 - \rho_r(a, b)\big] q_r(k), \\ \bar{v}_\varnothing(a, b) := \Pr(T > b \mid T > a) &= \sum_r \bar{\pi}_r(a) \rho_r(a, b). \end{aligned} \tag{11}$$

Setting $b = a + c$ gives a forecast over the next $c$ time units. The strict condition $T > a$ excludes the zero atom even at $a = 0$ and is well-defined for $p_0 < 1$. As the wait advances, only survival probabilities and mixture weights change; the history encoding, component parameters, and $q_r$ remain fixed. Sequential waiting updates agree with conditioning directly at the final waiting time. All these queries reduce to finite component sums without numerical time integration.

**Shared Computation Across Types and Requests** For $B$ horizons $b_1, \ldots, b_B$, collect component CDFs in $\mathbf{F} \in \mathbb{R}^{B \times R}$ and categorical probabilities in $Q \in \mathbb{R}^{R \times K}$, with $Q_{rk} = q_r(k)$. The positive-branch type–window probabilities form the matrix

$$\mathbf{V}_+ = \mathbf{F}\,\mathrm{diag}(\pi)Q, \qquad \mathbf{F}_{jr} = F_r(b_j/s). \tag{12}$$

The full probabilities include the factor $1 - p_0$ and zero-gap term in Eq. (9). The factorization in Eq. (12) separates temporal evaluation from type aggregation: $BR$ component-CDF evaluations serve all $K$ types, compared with $BKR$ for separate $R$-component banks.

To extend this reuse across requests, we decode the components once per history, evaluate uncached endpoints in batches, and reuse stored log-CDF and log-survival values. A new observed event requires a new history encoding and query state. Appendix A.4 provides the query algorithm, the derivation of waiting consistency, and stable tail calculations.

For $E$ unique endpoints, temporal evaluation costs $O(ERc_F)$, where $c_F$ is the cost of one component evaluation. Dense aggregation for $J$ requests returning all types costs $O(JRK)$. The shared bank reduces temporal evaluations across types, while the prepared state avoids repeated decoding and endpoint evaluation across requests.

# 4 Experiments

## 4.1 Experimental setup

**Data and Evaluation** We use Amazon, Stack Overflow (SO), MOOC, and Retweet, with 16, 22, 97, and 3 event types, respectively. Joint prediction uses all four datasets; learning controls and window queries use the first three. We use released EasyTPP splits (Xue et al., 2024) for Amazon, Stack Overflow, and Retweet, and a fixed user-level 70/15/15 split for MOOC. All models share the zero-atom/positive-density observation protocol. Code is included in the supplementary material.

**Models and Training** We compare STMix with LNM-Joint64 + summary (Waghmare et al., 2022), S2P2 (Chang et al., 2025), FlexTPP (Draxler et al., 2025), and AttNHP (Yang et al., 2022). STMix uses a GRU64 encoder and four spline components. Checkpoints minimize validation joint NLL. Structural and parameter-budget controls use dedicated runs, with training settings matched within each comparison. Experimental settings and control definitions are in Appendix B.

**Metrics and Reporting** We report test joint NLL per target in source time units, decomposed into time and conditional-mark NLL. Window log loss and Brier score evaluate next-type and no-event predictions. Unless stated otherwise, we report means and sample SDs over three training seeds.

## 4.2 Temporal sharing and joint learning

**Sharing Improves Test Prediction** The type-specific (untied) control assigns each type four independently parameterized temporal components; STMix shares four components across types with type-dependent weights. The encoder, density family, initial predictions, and training protocol are matched. STMix lowers test joint NLL in all nine dataset–seed pairs with 68.5–85.9% fewer total parameters (Figure 2). Mean test improvements are 0.0297, 0.0597, and 0.9307 on Amazon, Stack Overflow, and MOOC, respectively.

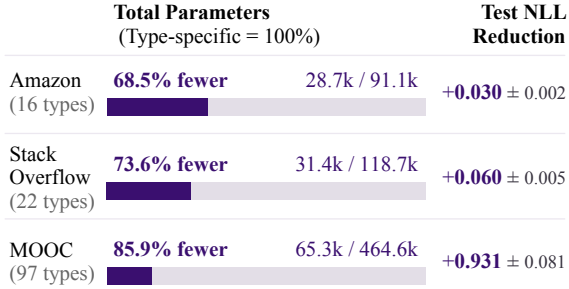



Figure 2: STMix: fewer parameters, lower test NLL in 9 paired runs. Reductions: type-specific − STMix.

**Parameter Budget and Dependence** On Amazon and MOOC, a separate matched experiment compares STMix with a compact type-specific head of similar parameter count and a time–mark independent control, starting from a common history-independent temporal density. The independent control factorizes the positive-gap joint density while retaining the time and mark marginal

families. STMix achieves lower mean test joint NLL than this control on both datasets, by 0.0390 on Amazon and 0.2014 on MOOC. At similar parameter counts, STMix improves over the compact type-specific model by 0.2825 on MOOC, whereas the compact model is better by 0.0826 on Amazon. These comparisons show that predictive behavior depends on how the distribution is organized, beyond its total parameter count.

**Marks Improve Temporal Learning** Using the main STMix runs as the reference, the posterior-stop comparison shows that direct mark-loss feedback can improve the temporal density (Table 1). On MOOC, retaining the feedback path lowers time NLL by 0.1605 and conditional-mark NLL by 0.0413, giving a joint improvement of 0.2018. Both terms improve in all three seeds on Stack Overflow as well; Amazon is approximately tied. Fixed and learned gates retain the shared temporal marginal family but replace the density-posterior weights with prescribed time-dependent or learned history-and-time-dependent weights. Their marked-window probabilities are evaluated by numerical integration to compare prediction and query cost.

Table 1: Weighting and feedback controls. Mean test joint NLL (↓) over three seeds.

| Model | Amazon | SO | MOOC |
|---|---|---|---|
| *Posterior feedback* | | | |
| STMix | $-0.7833$ | $\mathbf{2.2143}$ | $\mathbf{6.1074}$ |
| Posterior-stop | $\mathbf{-0.7840}$ | $2.2214$ | $6.3092$ |
| *Alternative conditional-mark weights* | | | |
| Fixed gate | $-0.7811$ | $2.2328$ | $6.3958$ |
| Learned gate | $-0.7829$ | $2.2149$ | $6.2274$ |

## 4.3 Joint and window predictive quality

**Complete Joint Predictors** STMix achieves lower mean joint NLL than LNM-Joint64 + summary, FlexTPP, and AttNHP on all four datasets (Table 2). S2P2 attains lower mean joint NLL on Stack Overflow, MOOC, and Retweet; the two models are close on Amazon and Stack Overflow. The larger Retweet gap comes from the time marginal: S2P2 has time NLL of 2.7950 versus 4.8644 for STMix, while STMix has lower conditional-mark NLL (0.7682 versus 0.8114; Appendix C.2).

Table 2: Test joint NLL per target in source time units (↓).

| Model | Amazon | Stack Overflow | MOOC | Retweet |
|---|---|---|---|---|
| LNM-Joint64 | $-0.6590 \pm 0.0067$ | $2.3664 \pm 0.0047$ | $7.2518 \pm 0.0278$ | $6.2250 \pm 0.0615$ |
| S2P2 | $-0.7793 \pm 0.0155$ | $\mathbf{2.2063 \pm 0.0097}$ | $\mathbf{5.8177 \pm 0.2308}$ | $\mathbf{3.6064 \pm 0.2284}$ |
| FlexTPP | $-0.5516 \pm 0.0046$ | $2.2583 \pm 0.0018$ | $7.0298 \pm 0.0888$ | $6.0972 \pm 0.0028$ |
| AttNHP | $-0.1921 \pm 0.2182$ | $2.2728 \pm 0.0196$ | $8.9635 \pm 0.0010$ | $6.5544 \pm 0.0371$ |
| STMix | $\mathbf{-0.7833 \pm 0.0021}$ | $2.2143 \pm 0.0007$ | $6.1074 \pm 0.0026$ | $5.6326 \pm 0.0081$ |

**Marked Window Prediction** Joint likelihood evaluates fit to individual time–mark observations; window scores assess the integrated probabilities used by finite-horizon forecasts. Horizons are the three training-gap quartiles; scores average over all targets and horizons. STMix achieves lower mean log loss and Brier score than LNM-Joint64 + summary on all three datasets. STMix and S2P2 have similar mean window log loss on Amazon and Stack Overflow. On MOOC, STMix reduces log loss and Brier score by 31.66% and 31.77%, reversing their joint-NLL ordering, with lower log loss at all three horizons (Figure 3). Full window scores appear in Appendix C.3.

**What Improves Within a Window** The MOOC advantage over S2P2 comes from both event-occurrence prediction and type prediction within the window (Figure 3). We decompose each window log loss into a binary occurrence loss and a type loss conditional on occurrence by the horizon, averaging both terms over the same full target set. Occurrence contributes more to the improvement at 4 and 21 seconds; at 447 seconds, the within-window type contribution is larger (0.3310 versus 0.1510). Both terms favor STMix in every seed at all three horizons. Appendix C.4 gives the exact decomposition and results for all five methods.

## 4.4 Efficient probability queries

**Quality and Cost Across Datasets** STMix completes the encoding-plus-query workload 3.53–5.82× faster than the learned gate across the three datasets (Figure 4). Its mean window log loss is lower on Amazon and Stack Overflow; on MOOC, the 5.82× speedup accompanies a 0.0042

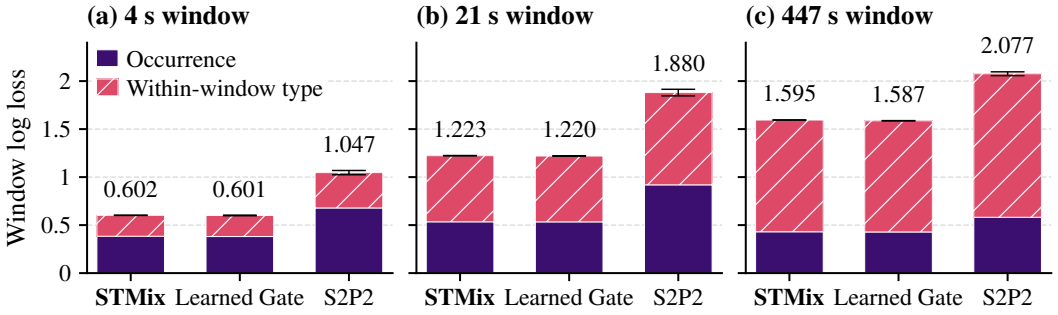



Figure 3: MOOC window log loss. Stacks show mean occurrence and within-window type losses; labels give totals, and error bars show sample SDs of total loss over three seeds.

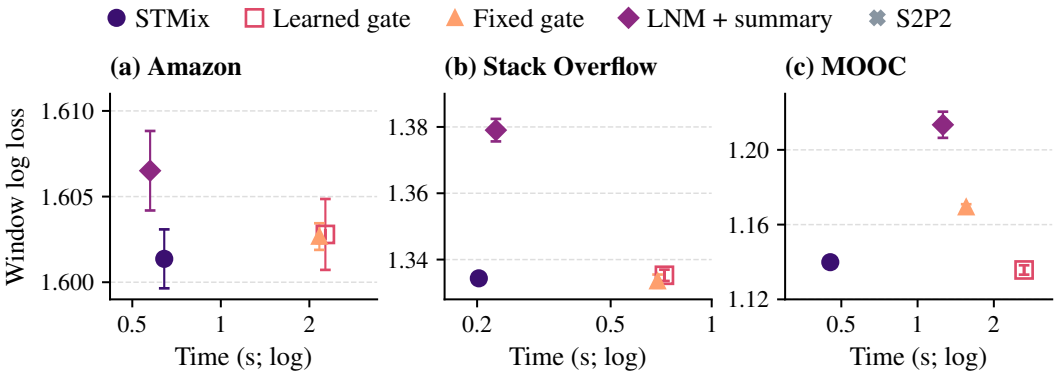



Figure 4: Window quality versus encoding-plus-query time for three horizons (three-seed means and sample SDs; dataset-specific axes). The MOOC S2P2 numerical-query adapter lies outside the displayed range: 2,768.8 s and 1.6679 log loss. Figure 10 shows the full range.

increase in log loss. LNM-Joint64 + summary is faster than STMix on Amazon, while STMix has lower window scores. Timings cover history encoding and all targets at three horizons on the same RTX 6000D. On MOOC, STMix evaluates three windows for all 58,226 targets in 0.4523 seconds, compared with 2.6329 seconds for the learned gate.

**Endpoint Reuse Across Request Types** Endpoint caching accelerates mixed queries over a vectorized reference; both paths decode components once per history. Using a fixed STMix checkpoint and 128 MOOC validation histories, we combine ordinary windows, adjacent conditional windows, and post-wait mark queries on grids of $B$ positive endpoints (Figure 5). At $B = 128$, caching reduces component-endpoint evaluations from 196,096 to 65,536 (66.6% fewer), and GPU workload time from 13.18 to 5.07 milliseconds. Timings include state preparation and exclude history encoding. Speedups range from 1.38 to 2.60× across the four workload sizes, with numerically equivalent predictions. The window-only control gains 1.00–1.04×, as its endpoints are already evaluated once. In a separate CPU control with a fixed query count and $B = 128$, increasing endpoint reuse from 0 to 66.5% raises speedup from 1.06× to 1.95×.

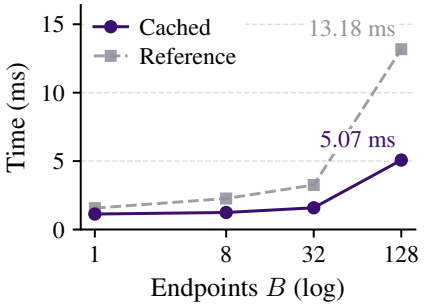



Figure 5: Endpoint reuse accelerates mixed queries (seven-repeat medians).

### 4.5 Reading the Shared Representation

**Four Components, All 97 Types** At one MOOC validation history, all 97 types combine the same four temporal components with type-dependent weights (Figure 6). Types placing more weight on the long-survival component accumulate probability later, while mixtures of components yield intermediate timing profiles. Thus, a small shared temporal bank supports type-specific timing variation.

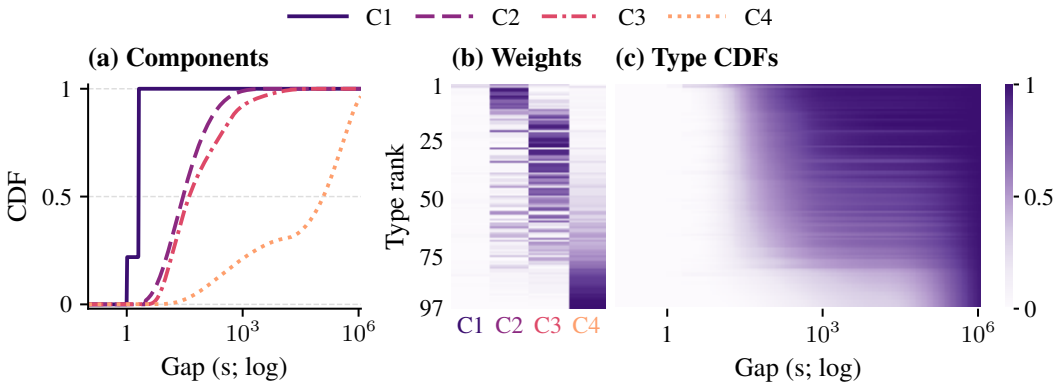



Figure 6: Four shared components organize all 97 MOOC types at one validation history. Type-conditioned weights (b) combine component CDFs (a) into type CDFs (c), conditional on a positive next gap. Both matrices share a color scale and rows sorted by predicted median gap.

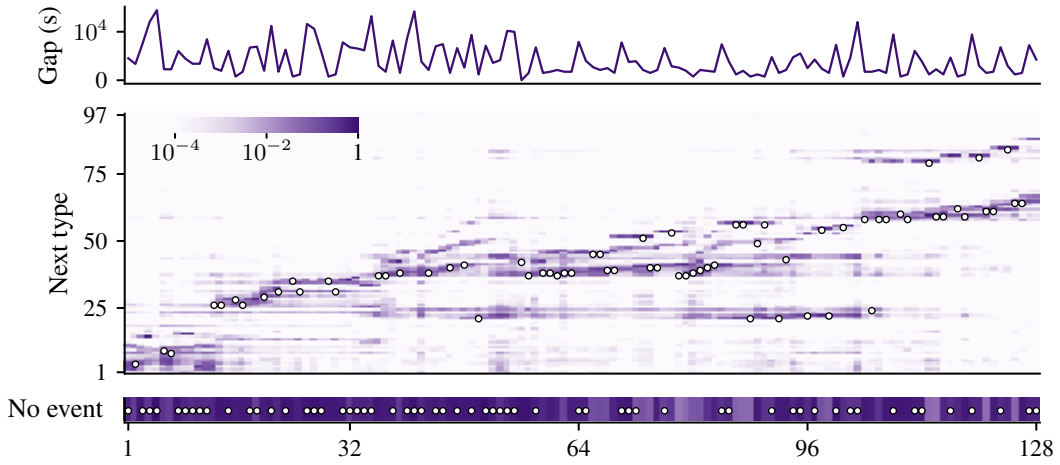



Figure 7: Forecasts along a real MOOC validation event stream. Colors show 21-second window probabilities for all 97 types and no event. The upper trace shows observed next gaps; outlined dots mark realized window outcomes. Colors use a log scale with a $10^{-4}$ display floor.

**Forecasts Along a Real Event Stream** Figure 7 follows 128 consecutive forecasts along a MOOC validation sequence. As new events arrive, the history-specific component bank and type weights update, shifting probability mass across event types. Between events, survival conditioning updates predictions while keeping the history encoding and component parameters fixed. On Amazon and MOOC validation data, this reweighting improves post-wait window log loss and Brier score over retaining the original mixture weights (Appendix C.5).

## 5 Discussion and Conclusion

We introduced STMix, whose shared temporal components support learning across event types and reusable probability computation. Type-dependent combinations retain distinct timing profiles, while joint likelihood lets marks shape shared temporal patterns. Tractable component CDFs enable marked-window and survival-conditioned forecasts without numerical time integration. Structural comparisons show better test likelihood with fewer parameters; query benchmarks demonstrate competitive window predictions and computational savings from reuse without changing predictions.

Joint likelihood and window scores can rank models differently, highlighting complementary aspects of forecast quality. Extending component reuse to multiple future events is a promising direction. Together, these findings show how shared temporal structure supports both joint learning and efficient probability queries.

### AI USE STATEMENT

We used generative AI tools, including ChatGPT and Codex, to assist with method development, mathematical derivations, experimental design, implementation, and result interpretation, as well as literature review, manuscript preparation, and visualization. The authors directed the research and manually reviewed the full manuscript, core code, figures, and tables. Verification included checks of mathematical and numerical consistency and comparisons with recorded experimental results. The authors take responsibility for the final content, including all AI-assisted contributions.

### ETHICS STATEMENT

This work studies probabilistic modeling of event sequences using publicly available benchmark datasets, without collecting new participant data. We use event times and categorical event types for offline evaluation and present aggregate results and illustrative validation-sequence examples. Applications involving personal behavioral data should respect privacy and data-use restrictions.

## REFERENCES

Emmanuel Bacry, Martin Bompaire, Stéphane Gaïffas, and Jean-Francois Muzy. Sparse and low-rank multivariate Hawkes processes. *Journal of Machine Learning Research (JMLR)*, 21(50): 1–32, 2020.

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 (ICLR)*, pp. 29610–29641, 2026.

Tanguy Bosser and Souhaib Ben Taieb. On the predictive accuracy of neural temporal point process models for continuous-time event data. *Transactions on Machine Learning Research (TMLR)*, 2023.

Tanguy Bosser and Souhaib Ben Taieb. Preventing conflicting gradients in neural marked temporal point processes. *Transactions on Machine Learning Research (TMLR)*, 2025.

Alex Boyd, Yuxin Chang, Stephan Mandt, and Padhraic Smyth. Probabilistic querying of continuous-time event sequences. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 10235–10251, 2023.

Yuxin Chang, Alex Boyd, Cao (Danica) Xiao, Taha Kass-Hout, Parminder Bhatia, Padhraic Smyth, and Andrew Warrington. Deep continuous-time state-space models for marked event sequences. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 38, pp. 162672–162713, 2025.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1724–1734, 2014.

Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1): 1–22, 1977.

Victor Dheur, Tanguy Bosser, Rafael Izbicki, and Souhaib Ben Taieb. Distribution-free conformal joint prediction regions for neural marked temporal point processes. *Machine Learning*, 113(9): 7055–7102, 2024.

Felix Draxler, Yang Meng, Kai Nelson, Lukas Laskowski, Yibo Yang, Theofanis Karaletsos, and Stephan Mandt. Transformers for mixed-type event sequences. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 38, pp. 114948–114980, 2025.

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 *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pp. 1555–1564, 2016.

Conor Durkan, Artur Bekasov, Iain Murray, and George Papamakarios. Neural spline flows. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, pp. 7509–7520, 2019.

Tilmann Gneiting and Adrian E. Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102(477):359–378, 2007.

Shukai Gong, Yiyang Fu, Fengyuan Ran, Quyu Kong, and Feng Zhou. TPP-SD: Accelerating Transformer point process sampling with speculative decoding. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 38, pp. 122170–122198, 2025.

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

Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87, 1991.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

Srijan Kumar, Xikun Zhang, and Jure Leskovec. Predicting dynamic embedding trajectory in temporal interaction networks. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pp. 1269–1278, 2019.

Sishun Liu, Ke Deng, Yongli Ren, Yan Wang, and Xiuzhen Zhang. Addressing mark imbalance in integration-free marked temporal point processes. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 38, pp. 61761–61792, 2025.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*, 2019.

David Lüdke, Enric Rabasseda Raventós, Marcel Kollovieh, and Stephan Günnemann. Unlocking point processes through point set diffusion. In *International Conference on Learning Representations (ICLR)*, pp. 82436–82454, 2025.

David Lüdke, Marten Lienen, Marcel Kollovieh, and Stephan Günnemann. Edit-based flow matching for temporal point processes. In *International Conference on Learning Representations (ICLR)*, pp. 111844–111864, 2026.

Hongyuan Mei and Jason M. Eisner. The neural Hawkes process: A neurally self-modulating multivariate point process. In *Advances in Neural Information Processing Systems (NIPS)*, volume 30, pp. 6754–6764, 2017.

Takahiro Omi, Naonori Ueda, and Kazuyuki Aihara. Fully neural network based model for general temporal point processes. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, pp. 2120–2129, 2019.

Aristeidis Panos. Decomposable Transformer point processes. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, pp. 88932–88955, 2024.

George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *Journal of Machine Learning Research (JMLR)*, 22(57):1–64, 2021.

Oleksandr Shchur, Marin Biloš, and Stephan Günnemann. Intensity-free learning of temporal point processes. In *International Conference on Learning Representations (ICLR)*, 2020a.

Oleksandr Shchur, Nicholas Gao, Marin Biloš, and Stephan Günnemann. Fast and flexible temporal point processes with triangular maps. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pp. 73–84, 2020b.

Oleksandr Shchur, Ali Caner Türkmen, Tim Januschowski, and Stephan Günnemann. Neural temporal point processes: A review. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 4585–4593, 2021.

Yujee Song, Donghyun Lee, Rui Meng, and Won Hwa Kim. Decoupled marked temporal point process using neural ordinary differential equations. In *International Conference on Learning Representations (ICLR)*, pp. 28158–28175, 2024.

Matthew A. Taddy and Athanasios Kottas. Mixture modeling for marked Poisson processes. *Bayesian Analysis*, 7(2):335–362, 2012.

Govind Waghmare, Ankur Debnath, Siddhartha Asthana, and Aakarsh Malhotra. Modeling interdependence between time and mark in multivariate temporal point processes. In *ACM International Conference on Information and Knowledge Management (CIKM)*, pp. 1986–1995, 2022.

Siqiao Xue, Xiaoming Shi, Zhixuan Chu, Yan Wang, Hongyan Hao, Fan Zhou, Caigao Jiang, Chen Pan, James Y. Zhang, Qingsong Wen, Jun Zhou, and Hongyuan Mei. EasyTPP: Towards open benchmarking temporal point processes. In *International Conference on Learning Representations (ICLR)*, pp. 52354–52375, 2024.

Chenghao Yang, Hongyuan Mei, and Jason Eisner. Transformer embeddings of irregularly spaced events and their participants. In *International Conference on Learning Representations (ICLR)*, 2022.

Simiao Zuo, Haoming Jiang, Zichong Li, Tuo Zhao, and Hongyuan Zha. Transformer Hawkes process. In *International Conference on Machine Learning (ICML)*, pp. 11692–11702, 2020.

# A METHOD AND QUERY DETAILS

We give the model parameterization, the recorded-event likelihood, the gradient calculation underlying joint learning, and the stable query algorithm. The notation follows Section 3: $T$ is a source-time gap, $U = T/s$ its normalized value, $K$ the number of marks, and $R$ the number of shared components.

## A.1 HISTORY ENCODER AND TEMPORAL PARAMETERIZATION

The encoder consumes one-hot marks and $\log(1 + u_j)$, with normalized observed gaps $u_j = (t_j - t_{j-1})/s$ and a zero input gap for the first event. A 64-unit GRU (Cho et al., 2014) followed by a learned 64-to-64 linear map gives $h_j$ for predicting event $j + 1$. STMix has no additional history-summary branch.

Three linear heads map $h$ to $R$ mixture logits, $RK$ categorical logits, and $Rd$ temporal parameters. Softmax normalization is over components for $\pi$ and over marks within each component for $q_r$. We use $R = 4$, $d = 16$, and history dimension $D = 64$. Counting the GRU's two bias vectors and the linear projections gives

$$P_{\text{shared}} = 3D(K + D + 3) + (D + 1)\big[D + R(1 + K + d)\big].$$

For $D = 64$, $R = 4$, and $d = 16$, this gives $452K + 21{,}444$ parameters. A type-specific temporal bank adds $(K - 1)Rd(D + 1) = 4160(K - 1)$, yielding the full-model counts in Figure 2. Sharing reduces the temporal head; the encoder and categorical head still scale with $K$.

Each temporal component has a location $\mu_r$, a scale $\sigma_r = 10^{-3} + \operatorname{softplus}(\eta_r)$, and 14 spline parameters. The spline has five bins on $[-5, 5]$. Widths and heights are each parameterized as $10^{-3} + (10 - 5 \cdot 10^{-3}) \operatorname{softmax}(\cdot)$. Four interior derivatives use $10^{-3} + \operatorname{softplus}(\cdot)$; boundary derivatives are one, and tails are the identity. We use the rational-quadratic construction of Durkan et al. (2019), within the normalizing-flow framework reviewed by Papamakarios et al. (2021). The transformation maps standardized log time *to* the normal base, giving the forward Jacobian in Eq. (4).

The temporal and mixture-logit projection weights start at zero, and the splines start at the identity. Main and sharing-control runs initialize component locations, scales, and mixture weights by deterministic one-dimensional $k$-means with four clusters on positive training log-gaps $\log u$. Centers start at the 12.5th, 37.5th, 62.5th, and 87.5th percentiles; Lloyd updates stop when the maximum center change is below $10^{-6}$ or after 50 iterations. Locations are cluster means, scales are within-cluster population standard deviations floored at $0.05$, and mixture weights use add-one-smoothed cluster counts. The budget-control initialization is specified in Appendix B.3.

The categorical heads initially share the same weight matrix, with component-specific bias offsets. Their initial logits are

$$\boldsymbol{a}_r(h) = W_0 h + \boldsymbol{b}_0 + \frac{r - 1}{3}\boldsymbol{v}_0, \qquad r \in \{1, 2, 3, 4\},$$

where $[W_0, \boldsymbol{v}_0] \in \mathbb{R}^{K \times 65}$ and $\boldsymbol{b}_0 \in \mathbb{R}^K$ are the weight matrix and bias of a single randomly initialized linear layer. Each component receives its own copy of $W_0$ and its offset bias; these parameters are trained independently thereafter.

Neural projections use single precision; spline transformations, normal log-CDFs, mixture reductions, and categorical normalization use double precision. Log-survival is evaluated directly as $\log \Phi(-z_r)$.

## A.2 NORMALIZATION AND THE RECORDED-EVENT OBJECTIVE

For fixed $h$, the transformation $u \mapsto g_r((\log u - \mu_r)/\sigma_r)$ is increasing from $(0, \infty)$ onto $\mathbb{R}$. Change of variables therefore gives $\int_0^\infty f_r(u \mid h)\, du = 1$. Since $\sum_k q_r(k \mid h) = 1$ and $\sum_r \pi_r(h) = 1$, Eq. (1) is a normalized joint distribution. Integrating or summing it gives the marginals and conditional distributions in Section 3.1. Including the zero-gap branch preserves normalization, since its mass is $p_0$ and the positive branch integrates to $1 - p_0$.

**Shared Structure and Time–Mark Dependence** At a fixed history, consider the positive-gap branch and a type with $m_k(h) > 0$. Then the weights in Eq. (2) satisfy $\alpha_{rk} \geq 0$ and $\sum_r \alpha_{rk} = 1$. Integrating that equation gives

$$\Pr(T \leq \tau \mid Y = k, T > 0, h) = \sum_{r=1}^{R} \alpha_{rk}(h) F_r(\tau/s \mid h).$$

Each type-conditioned CDF therefore lies in the convex hull of the same $R$ component CDFs. More generally, Eq. (12) expresses the positive-branch type–window matrix as a product of nonnegative $B \times R$ and $R \times K$ matrices, so its nonnegative rank is at most $R$. Type-dependent weights can yield distinct timing profiles within this common basis. For $R = 1$, the positive-gap joint density factorizes into $f_1(u \mid h)q_1(k \mid h)$ and time and type are independent. For any $R$, independence is also included when all $q_r$ coincide. Thus, multiple shared components permit dependence without requiring a separate temporal bank for every type. The basis and its weights vary with the history.

**Recorded Gaps and Time Units** For recorded gaps we use the measure $\delta_0 + d\tau$ on $[0, \infty)$ and counting measure on marks. Let $\mathcal{I}$ index valid prediction positions across all sequences, with the sequence index suppressed. At position $i$, history $h_i$ predicts the next mark $k_{i+1}$ and gap $\tau_i = t_{i+1} - t_i$. Each sequence contributes one target per event after its first. The loss is

$$\mathcal{L} = -\frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} \ell_i, \qquad \ell_i = \begin{cases} \log p_0 + \log q^0(k_{i+1} \mid h_i), & \tau_i = 0, \\ \log(1 - p_0) - \log s + \log p_+(\tau_i/s, k_{i+1} \mid h_i), & \tau_i > 0. \end{cases} \tag{13}$$

For each dataset, $s$ is the arithmetic mean of all training gaps after the first event of each sequence, including zero gaps. The mass $p_0$ is the fraction of these gaps that are zero, with no smoothing; both estimates are held fixed. We use $q^0 = q_1$, where component 1 is a fixed index, without reordering components by their learned locations. The term $-\log s$ converts positive densities to source time units. Recorded order and ties are preserved. The benchmark records do not provide a common terminal follow-up interval, so the objective scores observed next events without an additional terminal-survival term.

### A.3 Joint responsibilities and the posterior-stop intervention

The joint weights $\gamma_r$ are posterior probabilities of a latent component given the observed time–mark pair, the responsibility interpretation used in mixture-model expectation–maximization (Dempster et al., 1977). Here they appear in the exact gradient of the observed-data log likelihood; we train the conditional model directly by gradient optimization rather than alternating E- and M-steps.

For a positive target, the component log-scores $\xi_r = \log \pi_r + \log f_r(u \mid h)$ form $\boldsymbol{\xi} = (\xi_1, \ldots, \xi_R)^\top$, giving $w = \mathrm{softmax}(\boldsymbol{\xi})$ and $\mathcal{L}_{\mathrm{time}} = -\log \sum_r e^{\xi_r}$. Hence $\partial \mathcal{L}_{\mathrm{time}}/\partial \xi_r = -w_r$. Holding $q$ fixed at the component-score level, $\partial w_j/\partial \xi_r = w_j(\mathbf{1}_{j=r} - w_r)$ gives

$$\frac{\partial \mathcal{L}_{\mathrm{mark}}}{\partial \xi_r} = -\frac{\sum_j q_j(k \mid h) w_j(\mathbf{1}_{j=r} - w_r)}{\sum_j w_j q_j(k \mid h)} = w_r - \gamma_r. \tag{14}$$

Adding the two terms yields Eq. (7). For a parameter belonging only to temporal component $r$, the joint loss contributes $-\gamma_r \nabla \log f_r(u \mid h)$. These derivatives are local to component log-scores. For mixture logits $\boldsymbol{\beta}$ with $\pi = \mathrm{softmax}(\boldsymbol{\beta})$, the identity $\partial \log \pi_j/\partial \beta_r = \mathbf{1}_{j=r} - \pi_r$ gives

$$\frac{\partial \mathcal{L}_{\mathrm{time}}}{\partial \beta_r} = \pi_r - w_r, \qquad \frac{\partial \mathcal{L}_{\mathrm{mark}}}{\partial \beta_r} = w_r - \gamma_r, \qquad \frac{\partial \mathcal{L}_{\mathrm{joint}}}{\partial \beta_r} = \pi_r - \gamma_r. \tag{15}$$

Here the temporal and categorical component outputs are held fixed; shared-encoder gradients also include their other paths.

Posterior-stop replaces the mark loss during training by

$$\mathcal{L}_{\mathrm{mark}}^{\mathrm{stop}} = -\log \sum_r \mathrm{stopgrad}(w_r) q_r(k \mid h). \tag{16}$$

It detaches the density posterior $w$, not the joint responsibility $\gamma$. Its forward probabilities are unchanged for fixed parameters. For positive-gap targets, the time and mixture heads still receive the time loss; the categorical heads receive the mark loss; and the shared encoder receives gradients through both remaining paths. The intervention tests the direct feedback through the posterior, while retaining the common encoder. The zero-gap loss is unchanged.

**Algorithm 1** Reusable probability queries for STMix

**Require:** Fixed model and history encoding $h$, training-fitted $s > 0$, $p_0 \in [0, 1]$, and valid requests (Appendix A.4)

1: Decode temporal parameters, $\log \pi$, and $Q$ once; initialize endpoint cache $\mathcal{C}$.
2: **for** each batch of requests at this history **do**
3: Collect the unique required source-time endpoints absent from $\mathcal{C}$.
4: Batch-evaluate component log-CDFs and log-survivals at missing finite positive endpoints divided by $s$.
5: Assign analytic log-CDF and log-survival values at zero and infinity; cache the new factors.
6: Answer immediate mark and window requests by Eq. (9).
7: For post-wait requests, compute $\bar{\pi}(a)$ and, for windows, $\rho_r(a, b)$ from cached log-survivals.
8: Return $\bar{\pi}(a)^\top Q$ for post-wait marks or Eq. (11) for windows.
9: **end for**
10: Prepare a new state when the history, model parameters, device, or precision changes.

### A.4 Stable queries and state reuse

Algorithm 1 summarizes how a prepared state serves probability queries at a fixed history. Immediate windows allow $0 \le b \le \infty$. Post-wait queries require $0 \le a < \infty$, $a \le b \le \infty$, and $p_0 < 1$. Both endpoints are measured from the last observed event, so the residual window length is $b - a$. Immediate mark marginals require no temporal endpoint; post-wait mark marginals require only $a$.

The endpoint cache uses the boundary identities

$$F_r(0) = 0, \quad S_r(0) = 1, \qquad F_r(\infty) = 1, \quad S_r(\infty) = 0. \tag{17}$$

For source-time endpoints, define $L_r(a) = \log S_r(a/s)$. Stable conditional queries use

$$\begin{aligned} \bar{\pi}(a) &= \mathrm{softmax}\big(\log \pi + L(a)\big), \\ \rho_r(a, b) &= \exp\big(L_r(b) - L_r(a)\big), \\ 1 - \rho_r(a, b) &= -\operatorname{expm1}\big(L_r(b) - L_r(a)\big). \end{aligned} \tag{18}$$

This avoids subtracting nearly equal CDFs in the tail. All component survivals are positive for finite $a$. Conditional requests with $a = \infty$ or $p_0 = 1$ have zero-probability conditioning events and are rejected. At $b = a$, the no-event probability is one; $b = \infty$ gives the post-wait mark marginal.

**Consistency of Waiting Updates** For a further event-free wait with $0 \le a \le a' < \infty$, the posterior can be updated by

$$\bar{\pi}(a') = \mathrm{softmax}\big(\log \bar{\pi}(a) + L(a') - L(a)\big). \tag{19}$$

Substituting Eq. (10) cancels intermediate survival factors, recovering direct conditioning at $a'$. The implementation uses the original $\log \pi$ and cached $L(a')$ to avoid accumulated rounding error. Since $[1 - \rho_r] + \rho_r = 1$ and each row of $Q$ sums to one, the conditional $K + 1$ outputs are normalized. Immediate outputs are normalized after including the zero atom.

**Query Correctness and Shared Endpoints** In exact arithmetic, Algorithm 1 returns the marginals and conditional probabilities of the same decoded joint distribution. Initially the cache is empty. Each insertion stores its endpoint's component factors; retaining these entries preserves their values for subsequent requests at the same history. Substituting them into Eqs. (9) and (11) performs the corresponding marginalization and conditioning. The identities above establish normalization and consistency of repeated waits. For example, an immediate window ending at $b$, a conditional window $(a, b]$, and a type forecast after waiting to $a$, with $0 < a < b$, make four endpoint uses across the three requests but require only the two distinct endpoints $a, b$ in the shared cache. A new observed event requires a newly decoded state.

**How Waiting Changes Type Predictions** For fixed $h$ and finite $a > 0$, define the source-time component hazard

$$\lambda_r(a) = \frac{f_r(a/s \mid h)}{s S_r(a/s \mid h)}, \qquad \bar{\lambda}(a) = \sum_r \bar{\pi}_r(a) \lambda_r(a).$$

Because $dL_r(a)/da = -\lambda_r(a)$, differentiating the normalized survival weights gives

$$\frac{d\bar{\pi}_r(a)}{da} = \bar{\pi}_r(a)\big[\bar{\lambda}(a) - \lambda_r(a)\big]. \tag{20}$$

A component gains posterior weight when its current hazard is below the mixture hazard. Let $\bar{m}_k(a) = \Pr(Y = k \mid T > a, h) = \sum_r \bar{\pi}_r(a) q_r(k \mid h)$. The categorical components remain fixed during an event-free wait, so

$$\begin{aligned}\frac{d\bar{m}_k(a)}{da} &= \bar{\lambda}(a)\bar{m}_k(a) - \sum_r \bar{\pi}_r(a) q_r(k \mid h)\lambda_r(a) \\ &= -\operatorname{Cov}_{r\sim\bar{\pi}(a)}\big(q_r(k \mid h), \lambda_r(a)\big).\end{aligned} \tag{21}$$

Waiting thus favors types associated with currently lower-hazard components. The covariance can change sign as $a$ grows, so a type's probability need not be monotone. This update uses information from the absence of an event while keeping the history encoding and temporal parameters fixed.

**Computation and Storage** For a single fixed history, let $E$ count unique cached endpoints and $J$ count requests returning all $K$ type probabilities. After decoding the component parameters once, endpoint evaluation costs $O(ERc_F)$ and dense probability aggregation costs $O(JRK)$, where $c_F$ is the cost of one component evaluation. Endpoint caching avoids repeated component evaluations; type aggregation is still performed for each request. Cached state storage is $O(Rd + RK + ER)$, excluding the encoder and outputs. A waiting update at a cached endpoint costs $O(R)$, followed by $O(RK)$ for all mark probabilities. Endpoint evaluation is batched and chunked across histories and endpoints. Changes to the history, model parameters, device, or precision require a new state.

### A.5 Connections to related point-process models

**Temporal Bases and Shared Densities** Multivariate Hawkes processes model self- and cross-excitation between event types (Hawkes, 1971). Structured multivariate Hawkes models represent excitation kernels using temporal dictionaries and constrain the associated interaction coefficients. For example, Bacry et al. (2020) study sparse and low-rank regularization of the interaction tensor, with a fixed dictionary of decay kernels. This provides a useful connection between temporal basis functions and compact representations of cross-type interactions. In STMix, the shared objects are instead normalized next-gap densities decoded from the current history. Type-dependent weights combine these densities into distinct timing profiles. The nonnegative-rank bound in Appendix A.2 concerns a history-specific type–window probability matrix, rather than the interaction tensor of a Hawkes model. The shared densities' CDF and survival values also form common inputs to probability queries, linking the statistical representation to reusable computation.

**Product Mixtures and Conditional Forecasts** Input-dependent mixture weights are a classical mechanism for combining specialized predictors in mixtures of experts (Jacobs et al., 1991). In STMix, the observed history determines the mixture weights, and each component defines a product of a temporal density and a categorical mark distribution. Mixtures of continuous kernels and categorical distributions provide a statistical foundation for dependence between continuous observations and discrete marks (Taddy & Kottas, 2012). STMix applies this construction to a history-conditioned next-event distribution. For each observed time–mark pair, joint component responsibilities weight the learning gradients; during an event-free wait, survival probabilities reweight the components of the fixed predictor. Both operations use the history-conditioned component bank. This focus complements sequence-level approaches: triangular maps support parallel likelihood evaluation and sampling (Shchur et al., 2020b), and importance sampling supports broader future-event queries (Boyd et al., 2023). For the immediate next-event queries studied here, finite component sums give the requested probabilities directly.

**Sequence Generation and Sampling** Recent work expands point-process modeling through generation of entire point sets and sequences. Point Set Diffusion learns a reverse process on point sets in general metric spaces, supporting parallel point generation and conditional sampling (Lüdke et al., 2025). Edit-based flow matching constructs a continuous-time process of event insertions, deletions, and substitutions for temporal sequence generation (Lüdke et al., 2026). TPP-SD accelerates autoregressive sampling by using a smaller draft model to propose events and a target Transformer model

Table 3: Dataset sizes. Entries for each split are sequences / scored next-event targets. Amazon, Stack Overflow, and Retweet use released numerical time units; MOOC uses seconds.

| Dataset | Types | Train | Validation | Test |
|---|---|---|---|---|
| Amazon | 16 | 6,454 / 281,923 | 922 / 40,073 | 1,851 / 82,197 |
| Stack Overflow | 22 | 1,401 / 89,096 | 401 / 25,361 | 401 / 26,117 |
| MOOC | 97 | 3,281 / 268,851 | 703 / 55,823 | 704 / 58,226 |
| Retweet | 3 | 20,000 / 2,156,116 | 2,000 / 213,521 | 2,000 / 216,465 |

to verify them in parallel, with acceptance and correction steps preserving the target distribution (Gong et al., 2025). These developments improve generation flexibility and sampling efficiency, complementing the repeated probability evaluations studied here.

**Pretraining and In-Context Inference** FIM-PP pretrains on synthetic marked point processes and uses context sequences to infer history-dependent conditional intensities (Berghaus et al., 2026). It supports inference on new systems without parameter updates, as well as dataset-specific fine-tuning, and evaluates multi-event prediction. This approach shares inference capabilities across systems through pretraining. STMix shares temporal density components across event types within a history-conditioned predictor, using the same representation for joint learning and next-event probability queries.

# B Experimental Setup

This appendix specifies the experimental settings; Appendix C reports additional results.

## B.1 Data and observation protocol

Amazon, Stack Overflow, and Retweet use the released partitions from the EasyTPP dataset collection (Xue et al., 2024)[1]. Their dataset revisions are `508e8823`, `d356ab0d`, and `d2960029`, respectively. MOOC uses the SNAP student-action data (Kumar et al., 2019)[2]. We retain users with at least 20 recorded events, sort their IDs, and permute them with NumPy `default_rng` seed 20260912. The first $\lfloor 0.70N \rfloor$ eligible users form training, the next $\lfloor 0.15N \rfloor$ form validation, and the remainder form test, where $N = 4{,}688$. Original MOOC `TARGETID` values define the 97 marks; plotted type labels use one-based indices. Table 3 reports scored targets, excluding each sequence's first event. Source sequence order and ties are preserved without jitter or target truncation.

Table 4 gives the training-derived normalization and query horizons. Horizons are the 25th, 50th, and 75th percentiles of all training target gaps, including zeros. The training mean gap $s$ and zero-gap fraction $p_0$ are fixed before evaluation. All predictors use a common zero-atom/positive-density observation protocol and convert positive-gap densities back to source units. The first event is context only; no terminal-survival term is added because a common follow-up endpoint is unavailable. Aggregate losses are weighted by scored targets. We retain the numerical time units of the EasyTPP releases without conversion to physical units; MOOC times are in seconds. MOOC is split by user; the EasyTPP datasets retain their released sequence partitions, whose identifiers do not establish user-disjoint splits.

**Training-Set Profiles** Figure 8 shows the distribution of target types and their positive gaps. For the timing profiles, we order types by decreasing training frequency, break ties by type ID, and select ranks nearest the two endpoints and the one-third and two-thirds positions. Duplicate ranks are removed for Retweet. The empirical CDFs use all positive training gaps of each selected type, displayed over the dataset's 0.1st–99.9th positive-gap percentiles. These summaries describe variation across types and time scales, with observations pooled across histories.

Preliminary data inspection covered timestamp resolution, zero-gap frequencies, within-timestamp mark combinations, and file ordering across the Stack Overflow training, validation, and test splits

[1] `https://huggingface.co/easytpp`

[2] `https://snap.stanford.edu/data/act-mooc.html`

Table 4: Training-derived observation constants and window horizons in source time units. Values are rounded for display; evaluation uses the full-precision training estimates. Retweet is evaluated for joint prediction only.

| Dataset | Scale $s$ | Zero mass $p_0$ | $b_{25}$ | $b_{50}$ | $b_{75}$ |
|---|---|---|---|---|---|
| Amazon | 0.520952 | 0 | 0.01401233 | 0.72745467 | 0.76383479 |
| Stack Overflow | 0.889082 | 0.005141 | 0.18835449 | 0.53381348 | 1.17993164 |
| MOOC | 16,246.057924 | 0.037195 | 4 | 21 | 447 |
| Retweet | 2,596.137678 | 0.039994 | — | — | — |

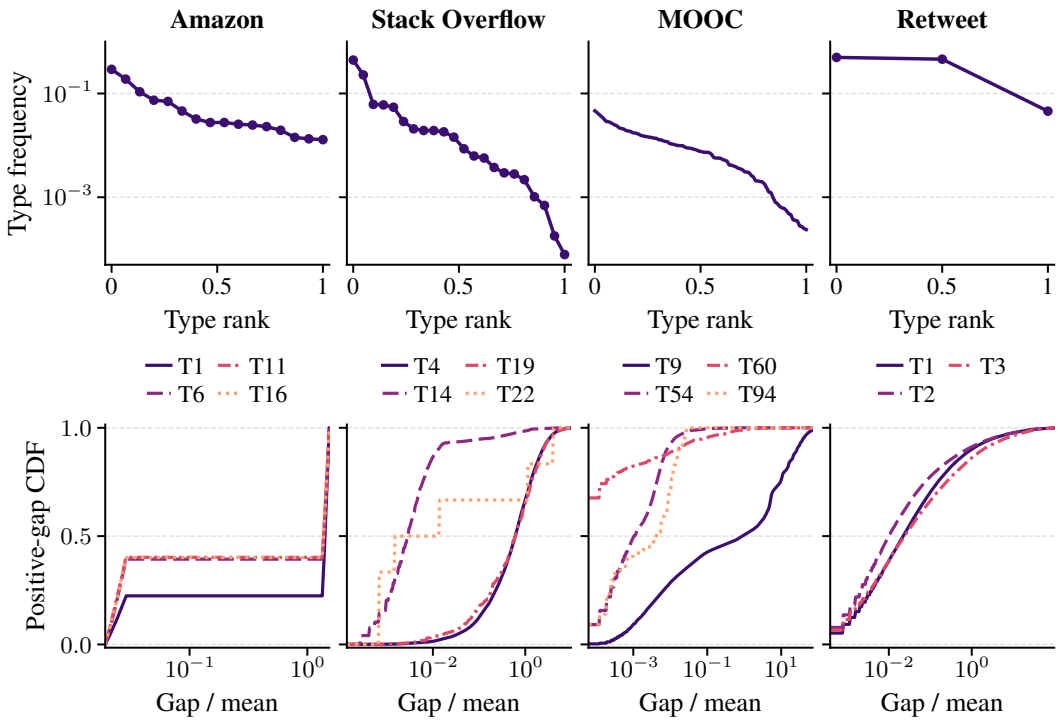



Figure 8: Training-set type frequencies and positive-gap timing profiles. Frequency rank runs from the most common type (0) to the least common (1). Time is divided by the training mean gap; T labels are one-based type IDs. The timing curves pool observations across histories.

and the unsplit MOOC source data. These inspections did not involve model training or comparisons of test predictive scores. Final model configurations and evaluation rules were subsequently fixed, and checkpoints were selected by validation joint NLL.

### B.2 Model configurations and training

STMix uses the architecture in Appendix A.1 on all four datasets. LNM-Joint64 uses a GRU64 encoder and 64 log-normal components per type, together with an additive history-summary projection into the 64-dimensional encoding. The summary contains per-type count, frequency, and recency features. This is the LNM-Joint64 + summary baseline throughout the paper.

Table 5 specifies the dataset-dependent FlexTPP and S2P2 architectures. FlexTPP uses five layers, feed-forward dimension 256, GELU activations, and normalization. S2P2 uses dropout 0.1, GELU, complex-valued states, relative time, post-normalization, and the simple-mark setting. AttNHP uses hidden size 32, time-embedding size 16, one layer, two heads, zero dropout, and no layer normalization. AttNHP and S2P2 use ten Monte Carlo integration samples per target during training; their final scoring settings are given below.

Table 5: Dataset-dependent baseline architectures. FlexTPP entries list positional-embedding capacity (`dim`), model width (`dim_m`), and per-head attention dimension (`dim_k`); dropout and spline bins also refer to FlexTPP. S2P2 lists hidden size, number of layers, and state size $P$. Shared settings are specified in the text.

| Dataset | FlexTPP dimensions | Dropout / bins | S2P2 hidden / layers / $P$ |
|---|---|---|---|
| Amazon | 250 / 48 / 12 | 0.40 / 11 | 64 / 2 / 128 |
| Stack Overflow | 250 / 48 / 12 | 0.46 / 10 | 32 / 3 / 32 |
| MOOC | 1010 / 48 / 12 | 0.40 / 11 | 64 / 2 / 128 |
| Retweet | 750 / 128 / 32 | 0.20 / 10 | 128 / 2 / 128 |

Table 6: Optimization settings. LR denotes the constant or peak learning rate; A, SO, M, and R denote Amazon, Stack Overflow, MOOC, and Retweet.

| Model / controls | Optimizer | LR / schedule | Decay | Clip |
|---|---|---|---|---|
| STMix and all learning controls | Adam | 0.003 / cosine | 0 | 1 |
| LNM-Joint64 + summary | Adam | 0.001 / constant | $10^{-5}$ | 1 |
| S2P2 | Adam | 0.01 / cosine | 0 | 1 |
| FlexTPP (A, M) | AdamW | 0.001 / OneCycle | 0.01 | 1 |
| FlexTPP (SO) | AdamW | 0.0004 / OneCycle | 0.01 | 1 |
| FlexTPP (R) | AdamW | 0.00015 / OneCycle | 0.01 | 1 |
| AttNHP | Adam | 0.001 / constant | 0 | 5 |

Decay is weight decay; clip is the global gradient-norm threshold. Adam uses $(\beta_1, \beta_2) = (0.9, 0.999)$ and $\epsilon = 10^{-8}$. AdamW uses the same $\epsilon$ and $\beta_2$, with $\beta_1$ varied by OneCycle.

**Configuration Sources and Selection** FlexTPP configurations are based on its paper-linked implementation, with dataset-specific type counts and the common observation protocol. For MOOC, we compared two configurations transferred from the Amazon and Stack Overflow settings using seed 17. The Amazon-derived configuration had lower validation joint NLL (7.0525 versus 7.2924) and was retained for seeds 29 and 43. Both candidates used 1,010 positional embeddings, twice the maximum training-sequence length of 505 events because each event contributes time and mark tokens. This changes positional capacity, not the model width. AttNHP uses the same EasyTPP-based architecture across datasets, with dataset-specific type counts. The S2P2 Retweet architecture follows the published hidden size, layer count, and state size, combined with the training and observation settings reported here.

**Optimization and Checkpoint Selection** Tables 6 and 7 give the optimization and execution settings. FlexTPP uses AdamW (Loshchilov & Hutter, 2019); the other models and learning controls use Adam (Kingma & Ba, 2015). Fixed-budget runs use 300 epochs on Amazon, Stack Overflow, and Retweet, with validation every five epochs. MOOC runs use 1,200 epochs, with validation every 20 epochs. Both schedules include epoch 0, giving 61 validation candidates per run. Selection minimizes validation joint NLL with earlier-checkpoint tie breaking. FlexTPP and AttNHP instead use the validation frequencies and early stopping specified in Table 7.

**Training Execution** Physical and effective batch sizes count sequences per forward/backward pass and per optimizer update, respectively. All valid targets contribute to the loss. For STMix and its learning controls, each update sums target losses over the effective sequence batch and divides by its number of sequences times the mean number of targets per training sequence. Gradient accumulation uses this same denominator for every microbatch, followed by one global-norm clip and one optimizer step. This normalization estimates the training-set target-average objective under uniform sequence sampling; evaluation divides by the actual number of scored targets.

The cosine schedules have a linear warmup over the first 1% of updates, followed by cosine decay to zero; warmup lengths are 78, 18, 156, and 237 updates for Amazon, Stack Overflow, MOOC, and Retweet. FlexTPP uses an epoch-stepped OneCycle schedule with 1,000 total steps, a 10%

Table 7: Training batches, budgets, and selection rules. Batch sizes count sequences; the last batch can be smaller. Dataset abbreviations follow Table 6.

| Model / run group | Data | Physical / effective | Budget / selection |
|---|---|---|---|
| Main STMix, fixed gate, posterior-stop, learned gate | A, SO, M | 256 / 256 | Fixed / 61 checks |
| Main STMix | R | 32 / 256 | Fixed / 61 checks |
| Sharing / type-specific | A, SO, M | 32 / 256 | Fixed / 61 checks |
| Budget / independence | A, M | 32 / 256 | Fixed / 61 checks |
| LNM-Joint64 + summary | A, SO | 256 / 256 | Fixed / 61 checks |
| LNM-Joint64 + summary | M | 16 / 256 | Fixed / 61 checks |
| LNM-Joint64 + summary | R | 32 / 256 | Fixed / 61 checks |
| S2P2 | A, SO | 256 / 256 | Fixed / 61 checks |
| S2P2 | M, R | 16 / 256 | Fixed / 61 checks |
| FlexTPP | A | 159 / 159 | 1,000 / patience 300 |
| FlexTPP | SO | 204 / 204 | 1,000 / patience 300 |
| FlexTPP | M | 8 / 159 | 1,000 / patience 300 |
| FlexTPP | R | 8 / 246 | 1,000 / patience 300 |
| AttNHP | A | 32 / 256 | 1,200 / patience 100 |
| AttNHP | SO | 256 / 256 | 1,200 / patience 100 |
| AttNHP | M | 8 / 256 | 1,200 / patience 100 |
| AttNHP | R | 16 / 256 | 1,200 / patience 100 |

Fixed budgets are 300 epochs on A, SO, and R and 1,200 on M, without early stopping. FlexTPP and AttNHP include epoch 0 and validate every epoch, except for the Retweet AttNHP seed-29 schedule described in the text. Maximum budgets and patience are measured in training epochs.

Table 8: Run groups for the learning comparisons. Each control uses the STMix reference from its own group. The final column reports MOOC test joint NLL (mean $\pm$ sample SD).

| Run group | Reported results | MOOC STMix |
|---|---|---|
| Main, weighting and feedback | Tables 1, 2, 12, and 13 | $6.1074 \pm 0.0026$ |
| Sharing / type-specific | Figure 2; Tables 9 and 10 | $6.1443 \pm 0.0965$ |
| Budget / independence | Table 11 | $6.6199 \pm 0.0367$ |

increasing phase, cosine annealing, an initial learning rate equal to one tenth of the maximum, and a final rate equal to $10^{-4}$ times the initial rate. Its first-moment coefficient cycles between 0.85 and 0.95.

For AttNHP on Retweet seed 29, validation was performed every epoch through epoch 383 and at absolute multiples of five thereafter, retaining earlier candidates. Patience remained 100 training epochs without improvement, checked at validation points; all other AttNHP runs used every-epoch validation. The run completed 1,200 epochs, including the final validation check, and selected epoch 1,175.

### B.3 Run groups and control definitions

**Run Groups** Table 8 identifies the three sets of STMix runs used in the learning comparisons. The weighting and feedback controls use the main STMix runs, also reported in Table 2, as their reference. Sharing and parameter-budget comparisons use their own matched STMix runs. On MOOC, the main and sharing-control runs share initialization and sequence orders but use different physical batch sizes (Table 7).

**Shared and Type-Specific Temporal Heads** The type-specific control replaces $f_r(u \mid h)$ by $f_{rk}(u \mid h)$ in the shared joint mixture, retaining the encoder, mixture and categorical heads, component family, and observation model. Each type has four independently parameterized spline compo-

nents. Copying the shared temporal head into every type gives the same initial predictions. Figure 2 gives full-model parameter counts and paired test gains; the count formula is in Appendix A.1.

**Similar Parameter Budgets** On Amazon and MOOC, we compare STMix with a compact type-specific head of similar parameter count and a time–mark independent control. The independent control uses $f_+(u \mid h)m_k(h)$ for the positive-gap joint density, where $m_k(h) = \sum_{r=1}^{R} \pi_r(h)q_r(k \mid h)$. It retains both marginal families and the STMix parameter count while removing time–mark dependence within the positive-gap branch. All three arms retain the same zero branch.

The compact type-specific head uses

$$\boldsymbol{\theta}_k(h) = \boldsymbol{b} + \mathbf{W}\big(\boldsymbol{e}_k \odot \mathbf{A}[h; 1]\big),$$

where $\mathbf{W} \in \mathbb{R}^{64\times d_{\text{fac}}}$, $\mathbf{A} \in \mathbb{R}^{d_{\text{fac}}\times 65}$, $\boldsymbol{e}_k \in \mathbb{R}^{d_{\text{fac}}}$, and $\boldsymbol{b} \in \mathbb{R}^{64}$. The output contains 16 parameters for each of four temporal components. The factor dimension $d_{\text{fac}}$ is fixed by the parameter budget: 28 for Amazon and 18 for MOOC. The resulting STMix/compact parameter counts are 28,676/28,640 and 65,288/65,260, respectively.

All four temporal components start with location $-0.5$, scale $0.8$, and identity splines. Each arm copies the encoder, categorical-head, and mixture-logit parameters from the same per-seed main initialization, including its training-derived mixture biases. The three arms therefore have identical initial joint predictions, with a common history-independent temporal density. For the compact head, $\mathbf{W} = 0$ and $\boldsymbol{b}$ encodes these temporal parameters; entries of $\mathbf{A}$ are uniform on $[-1/\sqrt{65}, 1/\sqrt{65}]$, and type factors are standard normal, using the training seed plus 190,919. The shared and independent arms use zero temporal projection weights with the same temporal biases.

**Weighting and Feedback** The fixed gate replaces the density posterior $w_r(u, h)$ in the conditional-mark distribution with the cubic Bernstein weights

$$g_r(u) = \binom{3}{r-1} z^{r-1}(1-z)^{4-r}, \qquad z = \frac{u}{1+u}, \quad r = 1, \ldots, 4.$$

Its joint prediction multiplies the shared time marginal by this conditional-mark distribution. The learned gate applies a $65 \to 32 \to 4$ network with a tanh hidden activation to $[h; \log u]$, adds its output to $\log g_r(u)$, and normalizes with softmax. Its output-layer weights and biases start at zero, matching the fixed gate initially; the network adds 2,244 parameters. Both gates retain the temporal and categorical component families. Posterior-stop preserves the STMix forward distribution and changes only the direct mark-loss gradient through $w$, as specified in Appendix A.3.

### B.4 Numerical evaluation and query workloads

**Likelihood and Uncertainty** Final likelihood evaluation reuses the validation adapters and selected checkpoints. STMix, LNM-Joint64, and FlexTPP use analytic density evaluation. AttNHP uses numerical integration with absolute and relative tolerances $10^{-5}$ and two successive confirmations, using Gauss–Legendre node counts $8, 16, 32, \ldots, 65{,}536$. Final AttNHP likelihood evaluation uses query chunks of eight. S2P2 uses eight independent 256-sample Monte Carlo likelihood estimates, with one predefined refinement to eight 1,024-sample estimates when the numerical 95% half-width exceeds 0.002. Numerical uncertainty and training-seed SD measure different sources of variation.

Training seeds are 17, 29, and 43. Numerical Monte Carlo repeats use predefined independent seeds, separate from the training randomness. All reported mean $\pm$ SD entries use the sample standard deviation across the three training seeds; SDs of paired gains are computed from the per-seed differences, not by combining the two models' marginal SDs.

**Window Scores and Numerical Adapters** For each fixed horizon, the outcome is the next mark if that event occurs by the horizon, and the no-event category otherwise. Log loss is the negative log probability assigned to this $K + 1$-class outcome. Multiclass Brier score sums squared errors across all $K + 1$ probabilities. Log loss and Brier score are strictly proper scoring rules (Gneiting & Raftery, 2007): in expectation, each is minimized by reporting the true outcome probabilities. Here the outcome space includes both the next-event types and the no-event category, so these scores

jointly assess event occurrence and type. They emphasize different discrepancies: log loss strongly penalizes assigning very low probability to an observed outcome, whereas Brier score measures squared probability error. Scores average over targets and the three training-defined gap-quartile horizons; checkpoints are not reselected for window quality. S2P2 window quality is evaluated on Amazon, Stack Overflow, and MOOC; its timing results cover MOOC.

STMix and LNM-Joint64 use analytic CDFs. The fixed and learned gates use their validation-fixed 64- and 128-node quadrature, respectively. For S2P2, we implement the three-horizon workload by integrating its competing-risk marked probabilities, with numerical settings selected on validation data. The adapter averages probability vectors from eight independent random stratifications. Amazon and Stack Overflow use 1,024 strata per segment for all three seeds; MOOC uses 8,192 for seeds 17/43 and 4,096 for seed 29. The Amazon and Stack Overflow quality evaluations extend the original MOOC evaluation using the existing validation-selected checkpoints and training-defined horizons. Their numerical settings were fixed after full-validation checks and before test evaluation.

Cross-model timing uses the same physical NVIDIA RTX 6000D GPU and covers all test targets and all three horizons. For each training seed, the median of three complete timed calls is recorded after warm-up; reported times are the mean and sample SD of these three seed medians. Query-only timing starts with encoded histories. Encoding-plus-query includes history encoding; neither includes data transfer or disk writes. S2P2 timings include all eight Monte Carlo repeats used to form each prediction and correspond to this adapter and its validation-selected numerical settings.

# C Additional Results and Analysis

We report structural and feedback comparisons, followed by complete predictive and computational results and validation examples of the learned representation. All model comparisons use the run groups and protocols in Appendix B.

## C.1 Temporal sharing and mark feedback

**Selected-Checkpoint Scores** Table 9 reports each model's training, validation, and test joint NLL at its checkpoint selected by validation joint NLL. All nine paired validation and test differences favor STMix. Training differences at these selected checkpoints favor STMix in all three Amazon and MOOC pairs, while Stack Overflow is approximately tied, with mixed directions across seeds. Selection epochs differ between models, so these training scores describe the selected predictors rather than a comparison at equal training progress.

**Final-Epoch Diagnostics** Table 10 reports training and validation differences at the end of the fixed budget. Type-specific models have lower training NLL, while STMix has lower validation NLL in all nine pairs. These diagnostics characterize the final training state; the predictive comparison above uses validation-selected checkpoints.

**Parameter Budget and Dependence** Table 11 reports the separately initialized controls defined in Appendix B.3. STMix achieves lower test joint NLL than the time–mark independent control on both datasets, supporting the value of modeling their dependence. At similar parameter counts, the compact type-specific head performs better on Amazon and STMix performs better on MOOC. Thus, a smaller parameter count alone does not explain the shared model's predictive behavior.

**Direct Mark Feedback and Alternative Weights** Table 12 gives the complete joint scores for the controls in Table 1; Table 13 separates time and conditional-mark NLL. Direct mark feedback improves both terms on MOOC and Stack Overflow, while Amazon is nearly tied. Figure 9 displays the paired contributions and their seed variability. The learned gate is competitive with STMix on Amazon and Stack Overflow, but has higher joint NLL on MOOC.

## C.2 Likelihood decomposition

For each scored target, joint NLL equals time-marginal NLL plus conditional-mark NLL, with the zero branch treated under the same observation measure. Table 14 reports the three-seed decomposition in source time units. Conditional-mark NLL uses the realized gap as a conditioning variable.

Table 9: Selected checkpoints in the matched structural comparison. Joint NLL per target in source time units. Each row reports Train, Validation, and Test at the same checkpoint, selected by the lowest validation joint NLL. Seeds are paired within each dataset; epochs may differ between models.

| Seed | Model | Epoch | Train | Validation | Test |
|---|---|---|---|---|---|
| *Amazon* | | | | | |
| 17 | STMix | 125 | −0.8078 | −0.7827 | −0.7833 |
| 17 | Type-specific | 75 | −0.7887 | −0.7514 | −0.7539 |
| 29 | STMix | 120 | −0.8052 | −0.7798 | −0.7821 |
| 29 | Type-specific | 70 | −0.7851 | −0.7490 | −0.7542 |
| 43 | STMix | 130 | −0.8137 | −0.7858 | −0.7850 |
| 43 | Type-specific | 85 | −0.7958 | −0.7536 | −0.7532 |
| *Stack Overflow* | | | | | |
| 17 | STMix | 300 | 2.2038 | 2.2113 | 2.2137 |
| 17 | Type-specific | 95 | 2.2147 | 2.2643 | 2.2695 |
| 29 | STMix | 275 | 2.2026 | 2.2110 | 2.2139 |
| 29 | Type-specific | 120 | 2.2010 | 2.2759 | 2.2796 |
| 43 | STMix | 285 | 2.2063 | 2.2137 | 2.2151 |
| 43 | Type-specific | 110 | 2.2043 | 2.2639 | 2.2725 |
| *MOOC* | | | | | |
| 17 | STMix | 1200 | 6.0641 | 6.2594 | 6.2551 |
| 17 | Type-specific | 120 | 6.9125 | 7.0581 | 7.0922 |
| 29 | STMix | 1200 | 5.9227 | 6.0739 | 6.0984 |
| 29 | Type-specific | 180 | 6.7939 | 7.0360 | 7.0722 |
| 43 | STMix | 1200 | 5.9087 | 6.0660 | 6.0793 |
| 43 | Type-specific | 160 | 6.8071 | 7.0274 | 7.0605 |

Table 10: Final-epoch structural diagnostics, reported separately from the selected-checkpoint comparison. Entries are type-specific minus STMix joint NLL at epoch 300 for Amazon and Stack Overflow and epoch 1,200 for MOOC. Positive values favor STMix. Test results use the validation-selected checkpoints in Table 9.

| Dataset | Seed | Train difference | Validation difference |
|---|---|---|---|
| Amazon | 17 | −0.0772 | +0.1207 |
| | 29 | −0.0813 | +0.1191 |
| | 43 | −0.0797 | +0.1172 |
| Stack Overflow | 17 | −0.1225 | +0.1328 |
| | 29 | −0.1122 | +0.1218 |
| | 43 | −0.1115 | +0.1119 |
| MOOC | 17 | −0.1980 | +1.1583 |
| | 29 | −0.0646 | +1.2783 |
| | 43 | −0.0806 | +1.2720 |

### C.3 Window quality and cross-model query cost

Table 15 reports window quality for all five models on three datasets under the protocol in Appendix B.4; Table 16 reports the completed same-device timing measurements. STMix has lower mean log loss and Brier score than LNM-Joint64 + summary on all three datasets. STMix and S2P2 have closely matched mean window log loss on Amazon and Stack Overflow; STMix has lower mean Brier scores on both, with a negligible difference on Amazon. The alternative gates remain competitive: the fixed gate has the lowest mean window scores on Stack Overflow and the learned gate on MOOC. STMix combines competitive window scores with 3.53–5.82× faster encoding-plus-query evaluation than the learned gate across the three datasets. Figure 10 shows the full quality–cost range, including S2P2 on MOOC.

Table 11: Parameter-budget and time–mark dependence controls. Test joint NLL, mean ± sample SD over three seeds, using the common initialization specified in Appendix B.3.

| Model | Amazon | MOOC |
|---|---|---|
| STMix | $-0.6047 \pm 0.0009$ | $6.6199 \pm 0.0367$ |
| Compact type-specific | $-0.6873 \pm 0.0052$ | $6.9024 \pm 0.0442$ |
| Independent | $-0.5657 \pm 0.0676$ | $6.8213 \pm 0.0548$ |

Table 12: Component weighting and direct mark-feedback controls. Test joint NLL (↓), mean ± sample SD over three seeds. The first block changes the posterior gradient path; the second changes the conditional-mark weighting rule.

| Model | Amazon | Stack Overflow | MOOC |
|---|---|---|---|
| *Posterior feedback* | | | |
| STMix | $-0.7833 \pm 0.0021$ | $\mathbf{2.2143 \pm 0.0007}$ | $\mathbf{6.1074 \pm 0.0026}$ |
| Posterior-stop | $\mathbf{-0.7840 \pm 0.0010}$ | $2.2214 \pm 0.0027$ | $6.3092 \pm 0.0453$ |
| *Alternative conditional-mark weights* | | | |
| Fixed gate | $-0.7811 \pm 0.0009$ | $2.2328 \pm 0.0032$ | $6.3958 \pm 0.0163$ |
| Learned gate | $-0.7829 \pm 0.0031$ | $2.2149 \pm 0.0021$ | $6.2274 \pm 0.0108$ |

### C.4 Decomposing window prediction quality

Let $v_k(b) = P(T \le b, Y = k \mid h)$, $p_b = \sum_k v_k(b)$, and $Z_b = \mathbf{1}[T \le b]$. With $r_k(b) = v_k(b)/p_b$, the multiclass window log loss has the exact decomposition

$$\ell_{\text{win}} = \underbrace{-Z_b \log p_b - (1 - Z_b)\log(1 - p_b)}_{\ell_{\text{occ}}} + \underbrace{-Z_b \log r_Y(b)}_{\ell_{\text{type|win}}}. \tag{22}$$

The type term is zero when no event occurs within the window. Both terms are averaged over all 58,226 targets, so their means add to the original window score. Here the type is conditioned on occurrence by $b$, whereas the likelihood decomposition in Appendix C.2 conditions on the realized gap.

We decompose the existing MOOC test predictions for all five methods, three seeds, and three fixed horizons, using the same checkpoints and horizons as the window evaluation. Figure 11 shows seed-matched differences; Table 17 retains absolute component scores. Against S2P2, both contributions favor STMix in all nine seed–horizon comparisons. The learned gate remains close to STMix, with small changes in both contributions.

### C.5 Predictive quality after an event-free wait

We evaluate the survival update using the three validation-selected STMix checkpoints on the complete Amazon and MOOC validation splits used for checkpoint selection. Waiting times are $a \in \{0, b_{25}, b_{50}, b_{75}\}$ and residual window lengths are $c \in \{b_{25}, b_{50}, b_{75}\}$, using the original training-gap horizons in Table 4. At each $a$, a target is eligible if its recorded gap satisfies $T > a$. The outcome is its next type when $T \le a + c$ and no event otherwise. We score all $K + 1$ outcomes by log loss and Brier score. The strict condition at $a = 0$ excludes zero-gap targets.

**Isolating Posterior Reweighting** The frozen-weight control retains the same encoder, temporal components, and categorical distributions, but replaces $\bar{\pi}(a)$ with the original $\pi$. It still updates each component's residual interval probability:

$$\widetilde{P}_k(a, c) = \sum_r \pi_r \big[1 - \rho_r(a, a + c)\big] q_r(k),$$

$$\widetilde{P}_{\text{none}}(a, c) = \sum_r \pi_r \rho_r(a, a + c).$$

Table 13: Time and conditional-mark contributions for the weighting and feedback controls (mean ± sample SD). Joint NLL is reported in Table 12.

| Dataset | Model | Time | Conditional mark |
|---|---|---|---|
| Amazon | STMix | $-2.6501 \pm 0.0018$ | $1.8668 \pm 0.0028$ |
| Amazon | Posterior-stop | $-2.6502 \pm 0.0012$ | $1.8662 \pm 0.0006$ |
| Amazon | Fixed gate | $-2.6515 \pm 0.0015$ | $1.8705 \pm 0.0009$ |
| Amazon | Learned gate | $-2.6511 \pm 0.0017$ | $1.8682 \pm 0.0031$ |
| Stack Overflow | STMix | $0.6987 \pm 0.0006$ | $1.5157 \pm 0.0013$ |
| Stack Overflow | Posterior-stop | $0.6997 \pm 0.0007$ | $1.5217 \pm 0.0034$ |
| Stack Overflow | Fixed gate | $0.6989 \pm 0.0008$ | $1.5338 \pm 0.0040$ |
| Stack Overflow | Learned gate | $0.6995 \pm 0.0015$ | $1.5154 \pm 0.0034$ |
| MOOC | STMix | $4.7268 \pm 0.0044$ | $1.3806 \pm 0.0025$ |
| MOOC | Posterior-stop | $4.8874 \pm 0.0428$ | $1.4219 \pm 0.0062$ |
| MOOC | Fixed gate | $4.8840 \pm 0.0149$ | $1.5118 \pm 0.0014$ |
| MOOC | Learned gate | $4.8606 \pm 0.0127$ | $1.3669 \pm 0.0019$ |

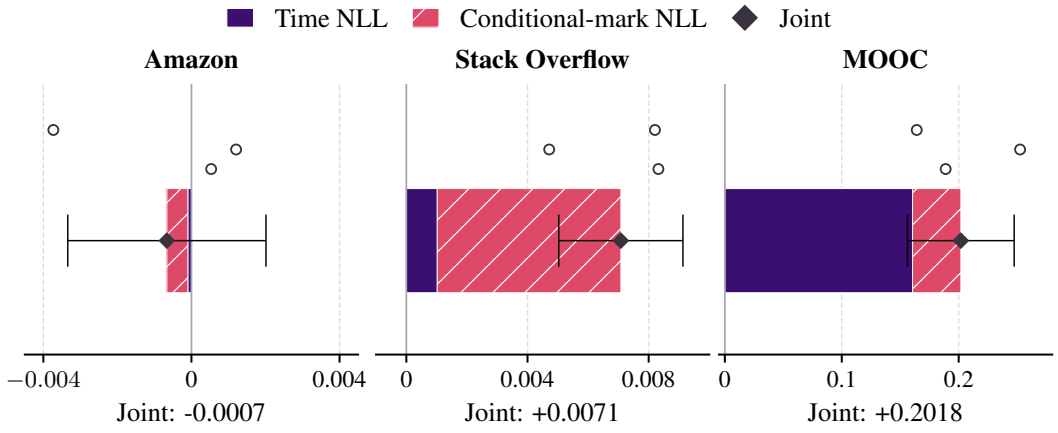



Figure 9: Posterior feedback and the two likelihood terms. Signed time and conditional-mark contributions to posterior-stop minus STMix test NLL sum to the joint difference. Diamonds and error bars show joint means and sample SDs of seed-matched differences; open circles show individual seeds. Positive differences favor STMix. Dataset-specific scales retain both the large MOOC improvement and the near tie on Amazon.

These probabilities sum to one, but omit the survival-induced change in component weights. This diagnostic control differs from the fixed-gate model in Table 1. At $a = 0$, the two predictors coincide; for positive waits, their difference isolates posterior reweighting.

Table 18 averages the three residual windows within each seed. For every positive waiting threshold, posterior reweighting lowers mean log loss and Brier score. Log-loss reductions range from 0.0779 to 0.0917 on Amazon and from 0.0488 to 0.1157 on MOOC. Both scores improve in all 54 nonzero-wait dataset–seed–wait–horizon combinations. The eligible set becomes smaller as $a$ increases, so model comparisons are paired within each waiting condition rather than across rows. Across the evaluated positive-wait conditions, survival-based posterior reweighting consistently improves window forecasts over retaining the original mixture weights.

### C.6 Endpoint reuse across request types

**Same-Model Endpoint Reuse** The GPU microbenchmark uses a separate, fixed set of 128 MOOC validation histories from seed 17, with $R = 4$ and $K = 97$. Identical precomputed history encodings

Table 14: Test likelihood decomposition for complete models. The mark term conditions on the realized gap. Entries are mean ± sample SD over three training seeds.

| Dataset | Model | Joint | Time | Conditional mark |
|---|---|---|---|---|
| Amazon | LNM-Joint64 + summary | $-0.6590 \pm 0.0067$ | $-2.5596 \pm 0.0062$ | $1.9006 \pm 0.0019$ |
| Amazon | S2P2 | $-0.7793 \pm 0.0155$ | $-2.6492 \pm 0.0108$ | $1.8699 \pm 0.0051$ |
| Amazon | FlexTPP | $-0.5516 \pm 0.0046$ | $-2.4354 \pm 0.0027$ | $1.8838 \pm 0.0019$ |
| Amazon | AttNHP | $-0.1921 \pm 0.2182$ | $-2.2023 \pm 0.2504$ | $2.0102 \pm 0.0552$ |
| Amazon | STMix | $-0.7833 \pm 0.0021$ | $-2.6501 \pm 0.0018$ | $1.8668 \pm 0.0028$ |
| Stack Overflow | LNM-Joint64 + summary | $2.3664 \pm 0.0047$ | $0.7189 \pm 0.0020$ | $1.6475 \pm 0.0031$ |
| Stack Overflow | S2P2 | $2.2063 \pm 0.0097$ | $0.6960 \pm 0.0031$ | $1.5103 \pm 0.0071$ |
| Stack Overflow | FlexTPP | $2.2583 \pm 0.0018$ | $0.7196 \pm 0.0017$ | $1.5387 \pm 0.0007$ |
| Stack Overflow | AttNHP | $2.2728 \pm 0.0196$ | $0.7563 \pm 0.0134$ | $1.5165 \pm 0.0062$ |
| Stack Overflow | STMix | $2.2143 \pm 0.0007$ | $0.6987 \pm 0.0006$ | $1.5157 \pm 0.0013$ |
| MOOC | LNM-Joint64 + summary | $7.2518 \pm 0.0278$ | $5.7230 \pm 0.0329$ | $1.5288 \pm 0.0054$ |
| MOOC | S2P2 | $5.8177 \pm 0.2308$ | $3.9348 \pm 0.2908$ | $1.8829 \pm 0.0607$ |
| MOOC | FlexTPP | $7.0298 \pm 0.0888$ | $5.6230 \pm 0.0965$ | $1.4068 \pm 0.0081$ |
| MOOC | AttNHP | $8.9635 \pm 0.0010$ | $6.8329 \pm 0.0159$ | $2.1306 \pm 0.0149$ |
| MOOC | STMix | $6.1074 \pm 0.0026$ | $4.7268 \pm 0.0044$ | $1.3806 \pm 0.0025$ |
| Retweet | LNM-Joint64 + summary | $6.2250 \pm 0.0615$ | $5.4137 \pm 0.0681$ | $0.8113 \pm 0.0347$ |
| Retweet | S2P2 | $3.6064 \pm 0.2284$ | $2.7950 \pm 0.2287$ | $0.8114 \pm 0.0034$ |
| Retweet | FlexTPP | $6.0972 \pm 0.0028$ | $5.3293 \pm 0.0027$ | $0.7680 \pm 0.0002$ |
| Retweet | AttNHP | $6.5544 \pm 0.0371$ | $5.7589 \pm 0.0287$ | $0.7955 \pm 0.0098$ |
| Retweet | STMix | $5.6326 \pm 0.0081$ | $4.8644 \pm 0.0081$ | $0.7682 \pm 0.0004$ |

Table 15: Window scores with training-seed variability. Entries are means ± sample SD; each seed score averages the same three windows and all test targets.

| Dataset | Model | Log loss | Brier |
|---|---|---|---|
| Amazon | LNM-Joint64 + summary | $1.6065 \pm 0.0023$ | $0.6402 \pm 0.0004$ |
| Amazon | Fixed gate | $1.6027 \pm 0.0008$ | $0.6395 \pm 0.0002$ |
| Amazon | Learned gate | $1.6028 \pm 0.0021$ | $0.6396 \pm 0.0003$ |
| Amazon | S2P2 | $1.6008 \pm 0.0038$ | $0.6395 \pm 0.0009$ |
| Amazon | STMix | $1.6014 \pm 0.0017$ | $0.6395 \pm 0.0003$ |
| Stack Overflow | LNM-Joint64 + summary | $1.3790 \pm 0.0034$ | $0.6054 \pm 0.0006$ |
| Stack Overflow | Fixed gate | $1.3335 \pm 0.0020$ | $0.5985 \pm 0.0003$ |
| Stack Overflow | Learned gate | $1.3353 \pm 0.0017$ | $0.5987 \pm 0.0003$ |
| Stack Overflow | S2P2 | $1.3343 \pm 0.0040$ | $0.5996 \pm 0.0010$ |
| Stack Overflow | STMix | $1.3343 \pm 0.0009$ | $0.5989 \pm 0.0003$ |
| MOOC | LNM-Joint64 + summary | $1.2134 \pm 0.0070$ | $0.5051 \pm 0.0020$ |
| MOOC | Fixed gate | $1.1693 \pm 0.0016$ | $0.4792 \pm 0.0006$ |
| MOOC | Learned gate | $1.1357 \pm 0.0025$ | $0.4720 \pm 0.0009$ |
| MOOC | S2P2 | $1.6679 \pm 0.0234$ | $0.6951 \pm 0.0053$ |
| MOOC | STMix | $1.1399 \pm 0.0009$ | $0.4742 \pm 0.0003$ |

are supplied to each path. Endpoint grids have size $B \in \{1, 8, 32, 128\}$; $B = 1$ uses the training median horizon, and larger grids use log-spaced positive endpoints over the fixed range from one quarter of 4 seconds to four times 447 seconds. Mixed requests use the same grid for windows, adjacent conditional windows, and post-wait mark probabilities (Figure 12).

Both paths decode once and vectorize evaluation. Without cross-request reuse, the reference requires $128 \cdot 4(3B - 1)$ component-endpoint evaluations; caching reduces this to $128 \cdot 4B$. Zero is handled analytically. Figure 5 reports median times over seven timing repetitions, including state preparation and excluding history encoding. Checks cover normalization, zero-gap boundaries, extreme tails, interval additivity, sequential versus direct waiting updates, and cache-history binding; the largest GPU output difference between the compared paths is $1.33 \times 10^{-15}$.

Table 16: Complete three-window workload time in seconds. Each seed uses the median of three full measurements; entries report the mean ± sample SD of the three seed medians. Transfer and disk I/O are excluded.

| Dataset | Model | Query only | Encoding + query |
|---|---|---|---|
| Amazon | LNM-Joint64 + summary | $0.5445 \pm 0.0077$ | $0.5749 \pm 0.0079$ |
| Amazon | Fixed gate | $2.1466 \pm 0.0084$ | $2.1629 \pm 0.0080$ |
| Amazon | Learned gate | $2.2501 \pm 0.0016$ | $2.2671 \pm 0.0018$ |
| Amazon | STMix | $0.6252 \pm 0.0062$ | $0.6421 \pm 0.0060$ |
| Stack Overflow | LNM-Joint64 + summary | $0.2235 \pm 0.0047$ | $0.2275 \pm 0.0012$ |
| Stack Overflow | Fixed gate | $0.6859 \pm 0.0011$ | $0.6898 \pm 0.0010$ |
| Stack Overflow | Learned gate | $0.7207 \pm 0.0029$ | $0.7230 \pm 0.0007$ |
| Stack Overflow | STMix | $0.1987 \pm 0.0011$ | $0.2024 \pm 0.0008$ |
| MOOC | LNM-Joint64 + summary | $1.2443 \pm 0.0026$ | $1.2588 \pm 0.0035$ |
| MOOC | Fixed gate | $1.5508 \pm 0.0036$ | $1.5555 \pm 0.0047$ |
| MOOC | Learned gate | $2.6225 \pm 0.0033$ | $2.6329 \pm 0.0055$ |
| MOOC | S2P2 | $2768.6482 \pm 957.6547$ | $2768.7630 \pm 957.6201$ |
| MOOC | STMix | $0.4441 \pm 0.0025$ | $0.4523 \pm 0.0023$ |

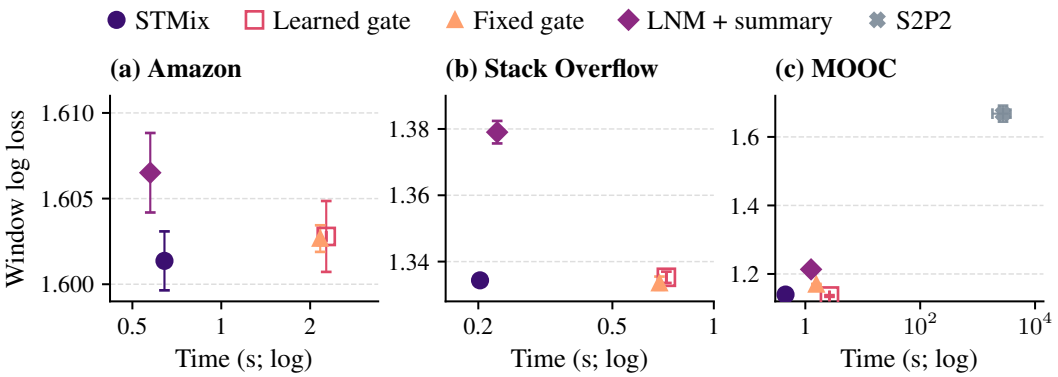



Figure 10: Complete-range quality–cost comparison for model–dataset combinations with completed timing measurements. Uncertainty follows Figure 4; the displayed range includes S2P2 on MOOC. Times cover history encoding and all three windows for every target. S2P2 uses the numerical-query adapter described in Appendix B.4, including all eight repeats. Each dataset has its own scales; every time axis is logarithmic.

**Controlling Endpoint Overlap on CPU** We vary overlap while holding each request family's size and output dimensions fixed, using the same frozen model and 128 validation embeddings as the GPU microbenchmark. This additional control runs on a local Intel Family 6 Model 183 CPU with two PyTorch threads; its timings are reported separately from the original RTX 6000D measurements. For $B \in \{8, 32, 128\}$, a $B+1$-point logarithmic grid from 1 to 1,788 seconds defines $B$ adjacent conditional windows. Ordinary windows initially use its right endpoints and post-wait mark queries its left endpoints. For nominal shared fractions $f \in \{0, 1/2, 3/4, 1\}$, the first $\lfloor fB \rfloor$ endpoints remain shared; the others are shifted by $\pm 0.17$ grid steps in log time for the two families. The small shifts change the edge of the covered range by less than one grid interval, and all conditional windows satisfy $a < b$.

Let $N$ sum the unique positive endpoints within each request family, and let $E$ count the unique endpoints across all families. Here $N = 3B + 1$, and the actual reuse fraction is $1 - E/N$. The reference deduplicates within each family; caching additionally deduplicates across families. Observed component-endpoint evaluation counts equal $128 \cdot 4N$ and $128 \cdot 4E$, respectively, with one call to each decoder head in both paths. Output probabilities agree within $10^{-12}$ for every setting.

At $B = 128$, full sharing reduces $E$ from 385 to 129 and component-endpoint evaluations from 197,120 to 66,048; the measured speedup grows from 1.06× at zero reuse to 1.95×. Full-sharing

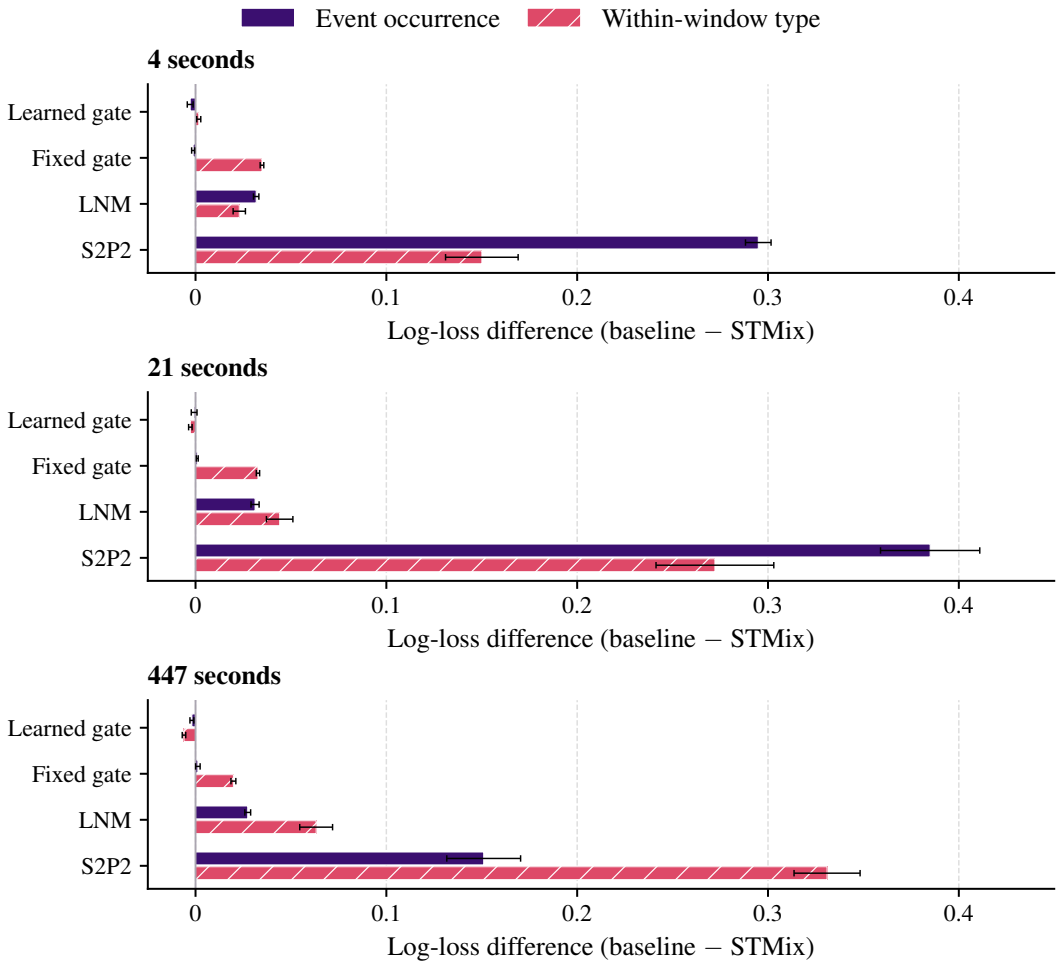



Figure 11: Contributions to MOOC window log-loss differences. Positive values favor STMix. Bars are mean event-occurrence and within-window type contributions; error bars are sample SDs of seed-matched differences. All three horizons and all four comparison models are retained. The two contributions sum to each model's total log-loss difference.

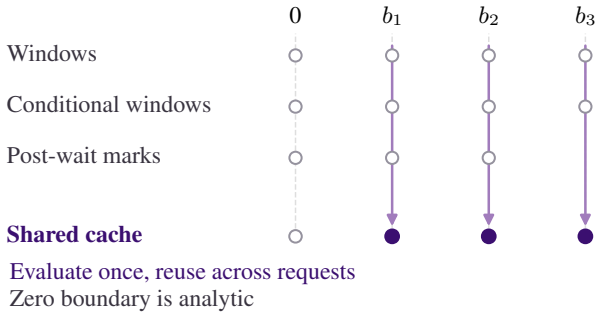



Figure 12: Endpoint reuse across query families. Open circles in the request rows indicate required endpoints; each positive endpoint is evaluated once for the shared cache. The zero boundary is handled analytically.

speedups across the three sizes are 1.69×, 2.22×, and 1.95×. The evaluation-count reduction is exact, while total time also includes state preparation, cache operations, and output aggregation, accounting for departures from the ideal endpoint-count ratio.

Table 17: MOOC window log-loss decomposition. Mean ± sample SD over three seeds; both terms average over all 58,226 targets and sum to the total.

| Horizon (s) | Model | Occurrence | Within-window type | Total |
|---|---|---|---|---|
| 4 | STMix | $0.3837 \pm 0.0005$ | $0.2181 \pm 0.0010$ | $0.6018 \pm 0.0012$ |
| 4 | Learned gate | $0.3809 \pm 0.0016$ | $0.2198 \pm 0.0018$ | $0.6007 \pm 0.0032$ |
| 4 | Fixed gate | $0.3825 \pm 0.0009$ | $0.2530 \pm 0.0017$ | $0.6354 \pm 0.0024$ |
| 4 | LNM + summary | $0.4155 \pm 0.0013$ | $0.2411 \pm 0.0023$ | $0.6565 \pm 0.0035$ |
| 4 | S2P2 | $0.6786 \pm 0.0072$ | $0.3682 \pm 0.0200$ | $1.0468 \pm 0.0221$ |
| 21 | STMix | $0.5343 \pm 0.0005$ | $0.6888 \pm 0.0012$ | $1.2232 \pm 0.0016$ |
| 21 | Learned gate | $0.5336 \pm 0.0013$ | $0.6861 \pm 0.0017$ | $1.2197 \pm 0.0028$ |
| 21 | Fixed gate | $0.5352 \pm 0.0005$ | $0.7216 \pm 0.0010$ | $1.2568 \pm 0.0014$ |
| 21 | LNM + summary | $0.5655 \pm 0.0018$ | $0.7329 \pm 0.0058$ | $1.2984 \pm 0.0076$ |
| 21 | S2P2 | $0.9193 \pm 0.0255$ | $0.9610 \pm 0.0317$ | $1.8803 \pm 0.0343$ |
| 447 | STMix | $0.4305 \pm 0.0003$ | $1.1642 \pm 0.0004$ | $1.5947 \pm 0.0007$ |
| 447 | Learned gate | $0.4286 \pm 0.0009$ | $1.1582 \pm 0.0008$ | $1.5867 \pm 0.0017$ |
| 447 | Fixed gate | $0.4317 \pm 0.0009$ | $1.1841 \pm 0.0011$ | $1.6158 \pm 0.0017$ |
| 447 | LNM + summary | $0.4578 \pm 0.0014$ | $1.2274 \pm 0.0085$ | $1.6853 \pm 0.0099$ |
| 447 | S2P2 | $0.5815 \pm 0.0191$ | $1.4952 \pm 0.0172$ | $2.0767 \pm 0.0199$ |

Table 18: Waiting-conditioned validation forecasts. Per-model log losses are three-seed means; gains are frozen minus STMix (mean ± paired sample SD). Each seed averages the three fixed residual windows. $N_a$ counts eligible targets, shared by both predictors.

| Wait $a$ | $N_a$ | STMix | Frozen | Log-loss gain | Brier gain |
|---|---|---|---|---|---|
| *Amazon* | | | | | |
| 0 | 40,073 | 1.6203 | 1.6203 | 0 | 0 |
| $b_{25}$ | 29,710 | 1.3547 | 1.4346 | $0.0799 \pm 0.0064$ | $0.0333 \pm 0.0028$ |
| $b_{50}$ | 19,799 | 1.5113 | 1.6031 | $0.0917 \pm 0.0055$ | $0.0380 \pm 0.0023$ |
| $b_{75}$ | 10,051 | 1.6744 | 1.7523 | $0.0779 \pm 0.0060$ | $0.0308 \pm 0.0028$ |
| *MOOC* | | | | | |
| 0 | 53,670 | 1.1405 | 1.1405 | 0 | 0 |
| $b_{25}$ | 41,569 | 1.1607 | 1.2096 | $0.0488 \pm 0.0008$ | $0.0247 \pm 0.0006$ |
| $b_{50}$ | 27,574 | 0.7728 | 0.8652 | $0.0924 \pm 0.0031$ | $0.0438 \pm 0.0018$ |
| $b_{75}$ | 13,934 | 0.3801 | 0.4958 | $0.1157 \pm 0.0055$ | $0.0490 \pm 0.0026$ |

### C.7 Learned components and waiting-conditioned predictions

These validation cases illustrate how shared components support type-specific timing and survival-conditioned updates. Figures 6 and 14 use the STMix checkpoint for MOOC seed 17, selected at epoch 1,200. Each case contains exactly the first 16 observed events of a validation sequence. Before computing predictions, we selected three distinct sequences whose last observed positive gap is nearest in log distance to each positive training-gap quartile (5, 24, and 482 seconds), breaking ties by sequence order. The resulting one-based validation sequence indices are 4, 7, and 296, with most recent gaps of 5, 24, and 484 seconds. These case-selection quartiles exclude zero gaps; the forecast horizons used in the window experiment are the previously fixed quartiles of all training target gaps. No next-event target enters case selection or the displayed history encoding.

The four highlighted types are the most frequent training targets: one-based IDs 9, 22, 8, and 10, with 12,459, 11,489, 10,473, and 9,693 targets, respectively. Each CDF is shown on a common logarithmic time range from one tenth of the minimum positive training gap to the 99.9th percentile of positive training gaps. The supplementary heatmaps give $\alpha_{rk} = \pi_r q_r(k) / \sum_j \pi_j q_j(k)$, conditional on a positive next gap. Component line styles identify decoder indices. Component parameters and weights vary with the history, so an index can represent different timing profiles across histories.

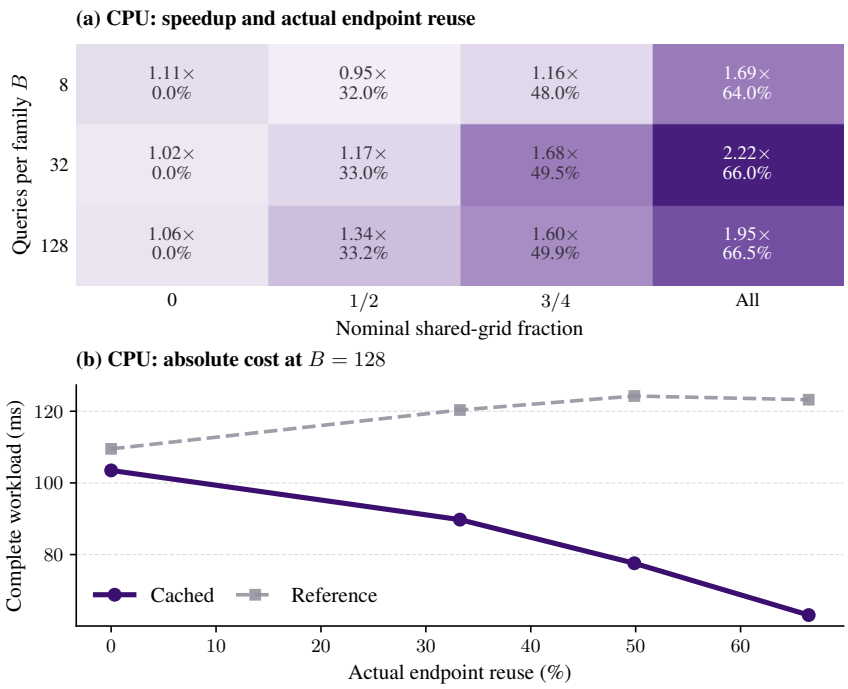



Figure 13: Endpoint-overlap control on CPU. (a) Each measured cell prints speedup and the actual reuse fraction; columns identify the nominal shared-grid setting. (b) Absolute times for $B = 128$. All paths include fresh state preparation and output aggregation. Seven repetitions alternate measurement order after two warmups. Small workloads can incur overhead: the $B = 8$ half-shared setting has a 0.95× ratio.

**All-Type CDF Atlas** Figure 6 expands History 7 to all 97 types using its archived mixture weights, categorical probabilities, and 1,000-point component-CDF grid. With $A_{kr} = \alpha_{rk}$ and $F_{\ell r} = F_r(t_\ell/s \mid h)$, the product $C = AF^\top$ contains $P(T \le t_\ell \mid Y = k, T > 0, h)$. We display 256 existing grid points selected uniformly by index. A single row permutation orders types by the first point on the full archived grid where their CDF reaches 0.5, with ties broken by type ID and non-crossing rows placed last. Both matrices retain this order. We use one fixed 0–1 CDF color scale, without row-wise rescaling. The atlas reconstructs the four highlighted type CDFs to within $10^{-12}$.

In these examples, the long-survival component receives increasing posterior weight by 447 seconds, raising the next-event probability of type 9. The update follows solely from observing that no event has occurred; all categorical component distributions and the encoded history remain fixed. The displayed probabilities are normalized over all 97 types.

For History 7, increasing the event-free wait from 0 to 447 seconds raises component 4's posterior weight from 26.6% to 82.4% and type 9's next-event probability from 23.7% to 70.0%. These are next-type probabilities without a forecast deadline; Figure 17 instead queries occurrence within a rolling 21-second window.

**Across Types, Waits, and New Events** Figure 16 adds Amazon, Stack Overflow, and Retweet cases from their frozen seed-17 models. In each dataset, we use 16 observed events and select the validation sequence whose latest positive observed gap is nearest in log distance to the median positive training gap, requiring at least 17 events and breaking ties by sequence index. This selects one-based sequences 590, 372, and 39, respectively. The three displayed types are the most frequent training targets; Other contains the remaining types. The middle column shows hypothetical event-free waits up to the training 75th-percentile positive gap. The last column instead incorporates the actual 17th event and recomputes the history encoding and temporal components.

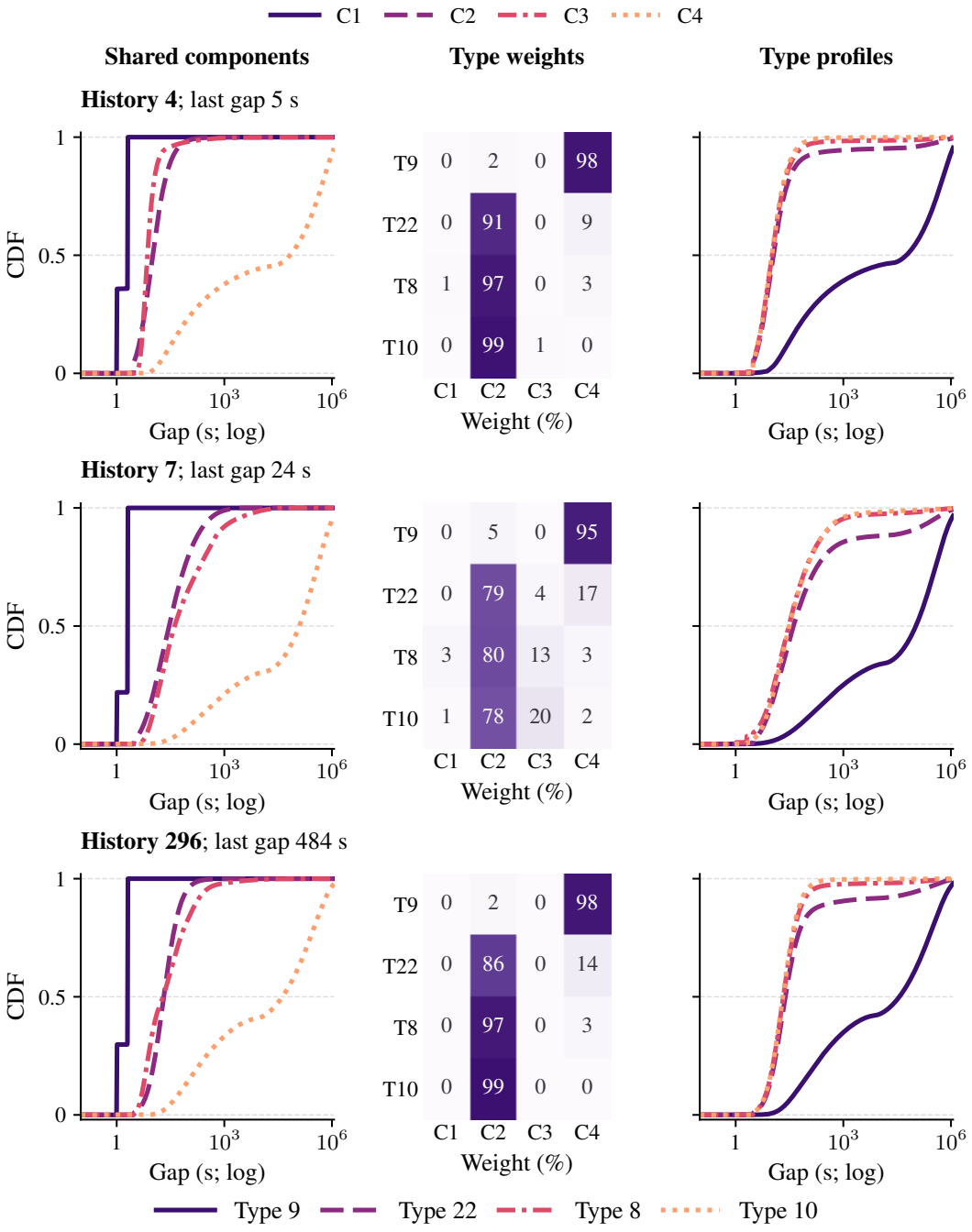



Figure 14: All three preselected validation histories. Each row shows temporal component CDFs, type-conditioned weights (percent), and the resulting type-conditioned CDFs. Rows of each weight matrix sum to 100% before rounding. Line styles identify component indices matching the matrix columns. Figure 6 expands History 7, the middle recent-gap setting, to all 97 types. All plotted timing profiles condition on $T > 0$.

The cases distinguish changes in type-specific combinations, survival weights, and the temporal bank itself. In the Retweet example, the largest change among the three type probabilities over the displayed wait is only 0.0056, while observing the new event changes the positive-gap time CDF by up to 0.296 on the displayed grid. Thus, a small response to waiting can coexist with a substantial

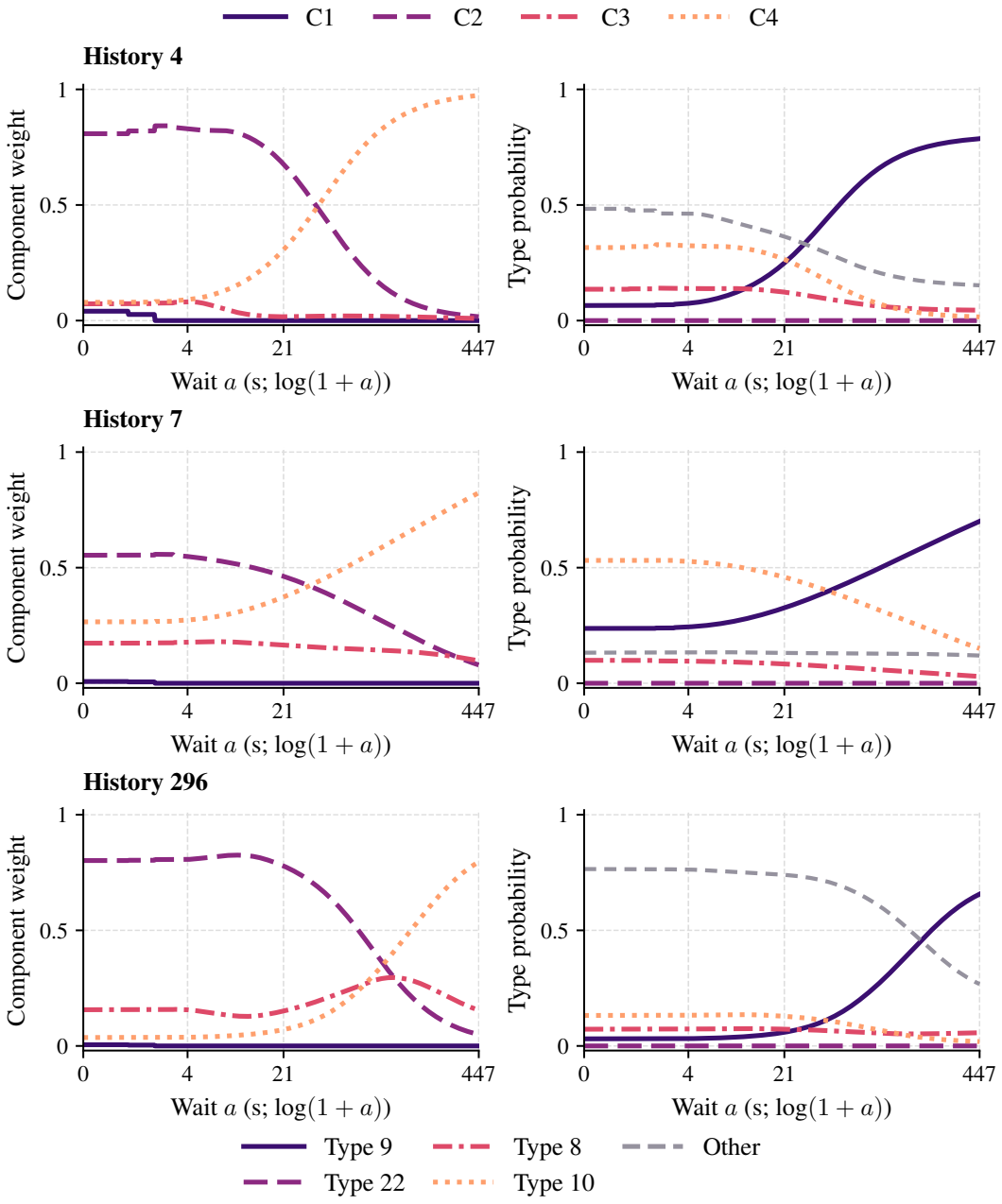



Figure 15: Predictions after an event-free wait for the same three histories as Figure 14, one history per row. Left: posterior component weights proportional to $\pi_r S_r(a/s)$. Right: next-type probabilities obtained by combining $q_r$ with those weights, including Other for the remaining 93 types. Curves use the archived dense CDF grid and include the original query settings $a = 0, 4, 21, 447$ seconds; the horizontal scale is $\log(1 + a)$. The $a = 0$ query already conditions on $T > 0$.

response to a newly observed event. The waiting derivatives in Eqs. (20) and (21) were also checked against central finite differences at the three training positive-gap quartiles for these fixed histories.

### C.8 Forecasts along a validation event stream

Figure 7 uses the same frozen MOOC seed-17 model and a separate, deterministically chosen validation sequence. Before forward evaluation, we select the first sequence in split order with at least

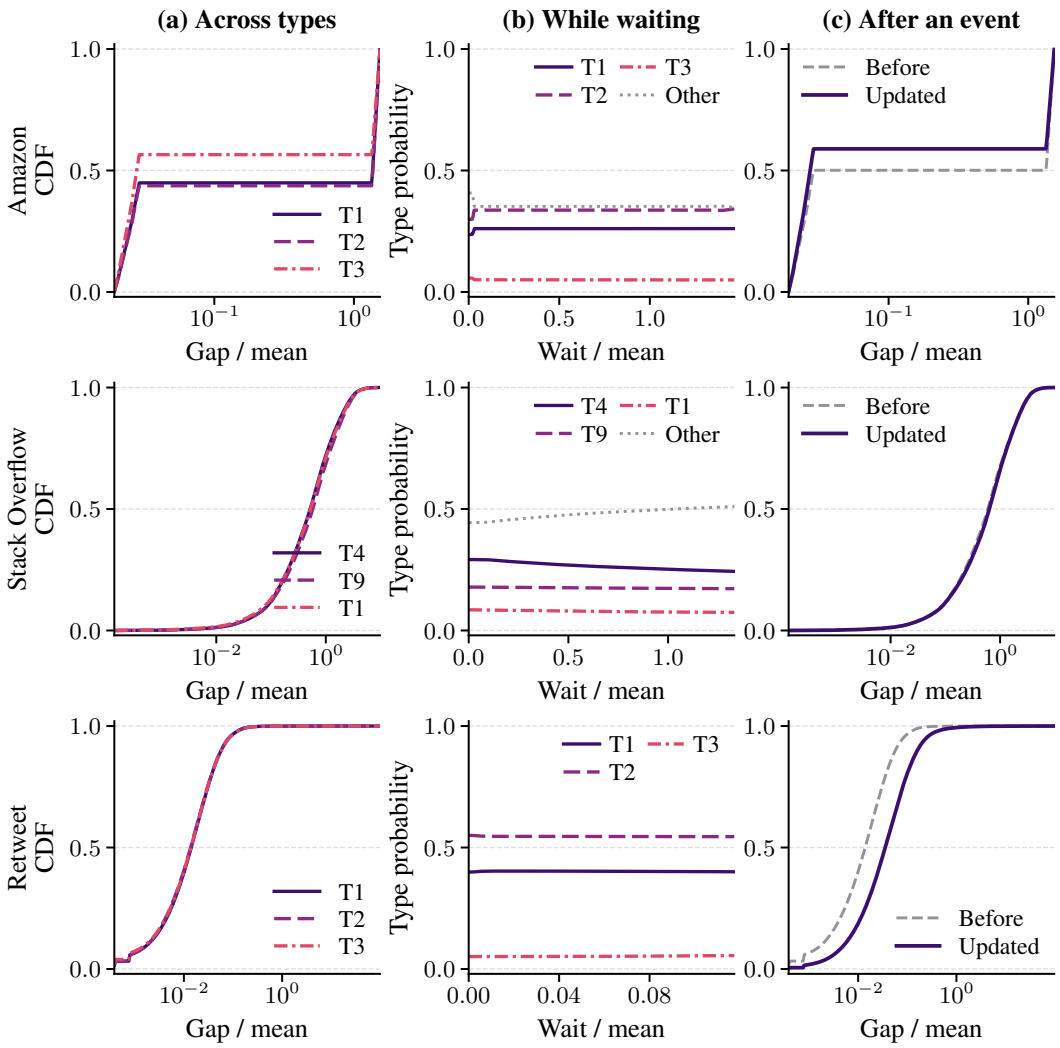



Figure 16: Three ways predictions vary, at preselected validation histories. (a) Type-conditioned positive-gap CDFs. (b) Next-type probabilities during a hypothetical event-free wait, with the history encoding fixed. (c) Positive-gap time CDF before and after the next observed event. Time is normalized by each dataset's training mean; type colors agree within each row.

129 events, giving one-based sequence 6 with 147 events. We show its first 128 prediction positions. Position $j$ uses only the first $j$ observed events to predict the next event, at the previously fixed 21-second horizon. The observed window outcome is the actual next type when its gap is at most 21 seconds and no event otherwise. The upper gap trace and outlined outcome markers are observational overlays; neither is supplied to that position's predictor. The heatmap retains all 97 marked probabilities and a separate no-event row, with a fixed logarithmic color scale from $10^{-4}$ to 1. Values below the lower limit are clipped only for display.

**A Real Event-Free Interval** Among prediction positions 16 through 128 whose next gap exceeds 21 seconds, we choose the gap closest in log distance to the fixed 447-second horizon, breaking ties by earliest position. This selects position 47, after 47 observed events, with an actual 447-second gap (Figure 17). The interval selection uses its observed duration, rather than prediction quality. Within this interval, 256 elapsed-wait values satisfy $0 \le a < 447$. At each value, we evaluate $P(a < T \le a + 21, Y = k \mid T > a, h)$ for every type and $P(T > a + 21 \mid T > a, h)$ for no event. The history encoding and component parameters remain fixed. At $a = 0$, conditioning on $T > 0$ already removes the zero atom, whereas the immediate window forecast in Figure 7 includes it.

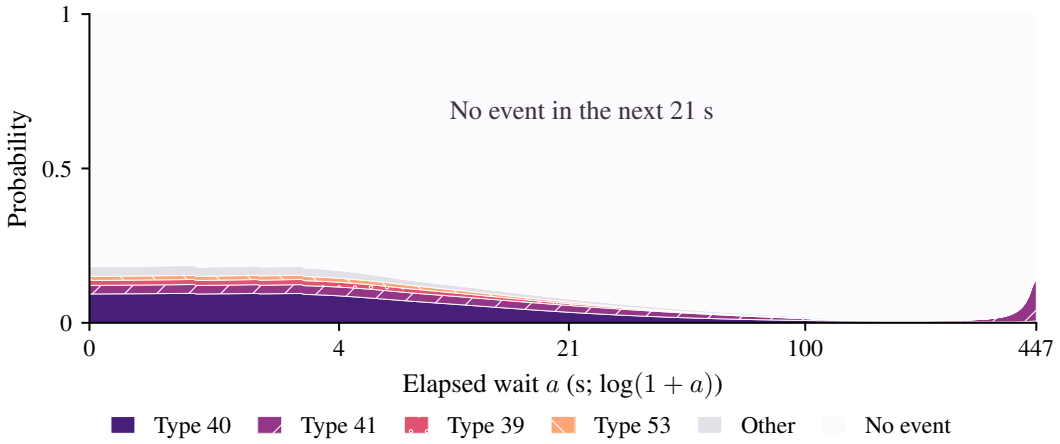



Figure 17: Waiting-conditioned forecasts during the 447-second gap at position 47 of the sequence in Figure 7. Bands give 21-second window probabilities conditional on $T > a$, with the history encoding and component parameters held fixed. The four displayed types have the largest marked-window probabilities at $a = 0$; Other sums the remaining 93 types.

For the stacked display, we select the four largest marked-window probabilities at the interval's origin under $T > 0$, breaking ties by type ID. This gives types 40, 41, 39, and 53; their order remains fixed for the entire wait, and Other sums the remaining 93 types. No future outcomes enter this type ranking. The six bands, including no event, sum to one.

The event-stream forecasts illustrate two complementary updates: new events change the history-specific state, while an event-free wait reweights its components. Queries within the same event-free interval reuse the history encoding and component parameters.