EDGE SELECTION FOR THE EFFECTIVE USE OF PIECEWISE-CONSTANT DISTRIBUTIONS AS NEURAL NETWORK OUTPUTS FOR EVENT PREDICTION

Anonymous authors
Paper under double-blind review

ABSTRACT

We study the output representation of a neural network used for next event prediction. We propose partitioning the time axis into a fixed set of intervals and having a neural network output a categorical distribution over them, which we map to a (mostly) piecewise-constant probability density. We present an optimization procedure that selects interval edges in order to maximize data likelihood under the representation. The representation is well suited to processes whose inter-event distribution is a mixture of smooth and sharply peaked components—a pattern we find common in event data recorded from real-world processes.

1 INTRODUCTION

Probabilistic modelling of event sequences with neural networks is the concern of neural temporal point processes (TPPs), reviewed by Shchur et al. (2021), Bosser & Ben Taieb (2023) and Lin et al. (2025). We focus on a narrow but consequential design choice: for models that take in an event history and predict the time until the next event, how should a network *represent* its prediction?

A prediction, in this context, is a probability density over $(0, \infty)$. Existing models reach such a density by parameterizing an intensity (hazard) function, or by parameterizing a distribution such as a mixture of log-normals. Our proposal takes the latter form. We have a neural network emit a categorical distribution over N intervals tiling $(0, \infty)$, and we interpret this output as a piecewise-constant density function (with a single exponential tail for the final unbounded interval). The effectiveness of this approach depends heavily on the choice of interval edges. We present a dynamic programming optimization of the intervals that maximizes data likelihood under an empirical distribution. We will show that outputting a piecewise-continuous model over these edges beats existing output heads at next event prediction on 14 standard datasets in terms of log-likelihood.

This approach will be shown to be particularly effective for processes that exhibit a mixed distribution containing continuous and discrete parts. The times between taxi pickups in New York City obtained by Whong (2014) have such a distribution—the distribution is relatively smooth but is punctuated by regular peaks (see Figure 1). Appendix C.1 presents the histograms of 14 real-world datasets, demonstrating that this phenomenon is common.

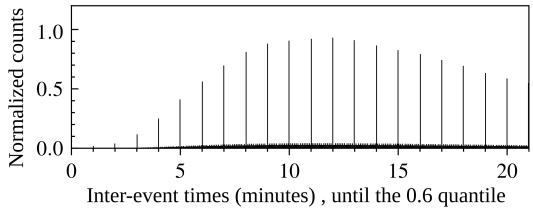


Figure 1: Histogram of inter-event times (< 0.6 quantile) for the NYC taxi dataset (Whong, 2014).

The next section establishes the next event prediction problem; Section 3 describes the form of the piecewise-constant density head we use; Section 3.1 gives the edge optimization procedure; Sections 5 to 6 cover experiment setup and results; and ablations in Section 7 test properties of the output representation. Beyond supporting the effectiveness of our approach, the results hint that next event prediction on datasets recorded from real-world processes should consider a resolution of interest, possibly the precision at which events are recorded. This and other points are discussed in Section 8 and Section 9.

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

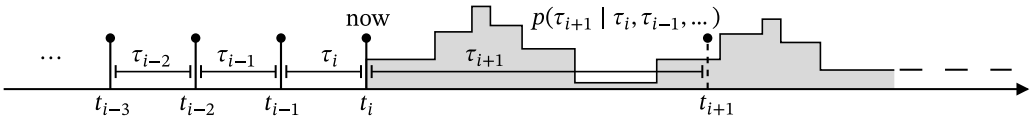


Figure 2: The next event prediction task. Given a history of inter-event times $\tau_1, \tau_2, \ldots, \tau_i$ up to the present, the task is to emit a probability distribution $p(\tau_{i+1} \mid \tau_i, \tau_{i-1}, \ldots)$ for the time until the next event, τ_{i+1}. This paper considers a density that is piecewise-constant over bins of non-uniform width, with an exponential tail over the final infinite interval.

2 Next event prediction

Let $(t_i)_{i=0}^T$ be a sequence of non-negative strictly increasing reals representing event times. Denote inter-event times as $\tau_i = t_i - t_{i-1}$. In this work, all models will receive a history of inter-event times, $\mathcal{H}_i = (\tau_i, \tau_{i-1}, \ldots, \tau_1)$. The task is: given history $\mathcal{H}_i$, produce a probability density function over $(0, \infty)$ representing the time until the next event, τ_{i+1}. Figure 2 illustrates the task and a model that outputs a piecewise-constant density function. Models are scored by the log-likelihood they assign to inter-event times of a held-out test set. For a model with parameters θ producing a density function p_θ, this log-likelihood score is given by:

$$\ell\ell \;=\; \sum_{i=1}^{T} \log p_\theta(\tau_i \mid \mathcal{H}_{i-1}), \tag{1}$$

summed across sequences, or if reported as a mean, it is a mean over all events. For all models we consider, the likelihood is also a training signal (a negative log-likelihood loss).

We do not consider models of event counts within intervals, such as those from Lüdke et al. (2023) and Li et al. (2024); the probability spaces are not directly comparable as their likelihood is defined over a different observation space.

3 Output representation: piecewise-constant density function

We build a (mostly) piecewise-constant probability density function over $(0, \infty)$ in two steps. A preview of outputs emitted by a model using this representation is shown in Figure 4 (top row).

Step 1: categorical output. A network emits N logits $\mathbf{z} = [z_1, ..., z_N]$ which a softmax maps to a categorical distribution, $\mathbf{m} = \mathrm{softmax}(\mathbf{z})$ over N intervals (see Figure 3 below). Write m_i for the probability mass assigned to the i-th interval. These N intervals will tile $(0, \infty)$. Of the N intervals, $N - 1$ have finite length and one (the last) has infinite length. The intervals have strictly increasing edges $a_1, a_2, \ldots, a_N$. The first interval (a_1, a_2) begins at zero; the second interval is $[a_2, a_3)$, and so on until the final interval, $[a_N, \infty)$. Choosing the edges is the topic of Section 3.1.

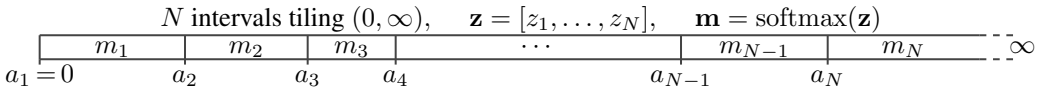


Figure 3: The intervals, logits and probability masses defining the categorical output.

Step 2: mass to density. Each finite interval gets a constant density. The final interval, being infinite in length, cannot have a constant non-zero density; instead, it gets an exponential decay.

The density function $p(\tau)$ is then defined as:

$$p(\tau) = \begin{cases} \frac{m_i}{w_i} & \tau \in [a_i, a_{i+1}), i < N \\ m_N \lambda e^{-\lambda(\tau - a_N)} & \tau \in [a_N, \infty), \end{cases} \tag{2}$$

where λ is the decay of the final interval's density, and w_i is the interval width, $w_i = a_{i+1} - a_i$.

The decay, λ, of the exponential distribution is fit by maximum likelihood to the tail end of the training set inter-event times; in this work, we fit λ using the largest 0.5% of training set inter-event times. The final (infinite) interval prevents the density function from being considered wholly

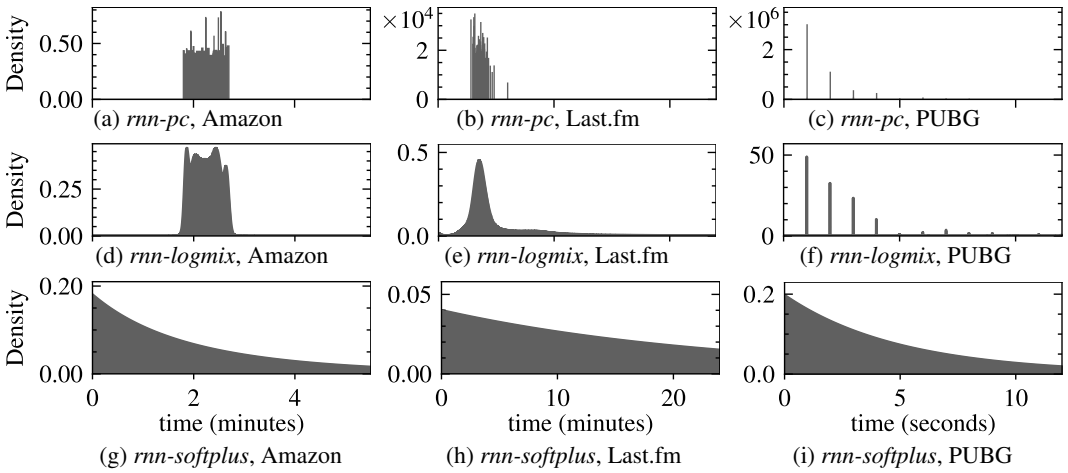


Figure 4: Probability distributions corresponding to outputs from 3 trained models with different heads: a piecewise-constant density, a mixture of lognormals density and a softplus hazard; these are labelled as *rnn-pc*, *rnn-logmix*, and *rnn-softplus*, respectively. See Section 5 for model and head details. Each model is given the same input history from three datasets: Amazon, Last.fm and PUBG (datasets are introduced in Section 5.3 and histograms of inter-event times are in Appendix C.1).

piecewise-constant; however, as will be shown, the final interval will be designed to rarely contain events and will not significantly contribute to the likelihood score.

3.1 Choosing interval edges

Before placing edges, we must decide how many are needed. We use $N = 128$ for the main results and examine a range of values in Section 7.

For placing the edges, we investigate the following approaches (acronyms are in parentheses):

1. equal-width intervals (*wpc*)
2. equal-quantile intervals (*qpc*)
3. intervals that maximize likelihood for an unconditional model (*pc*)

Equal-width and equal-quantile intervals are standard techniques for histogram construction. They were also studied by Kvamme & Borgan (2021) in the context of survival analysis with static covariates. There is a long history of data-dependent construction of variable-width histograms (Kanazawa, 1992; Rozenholc et al., 2010; Scott, 2010; Kontkanen & Myllymäki, 2007; Simensen et al., 2026). We use the third approach for the main results, and Appendix B.1 covers the other two.

The second and third approaches exploit narrow intervals to capture peaks in the distribution. For the main results, minimum widths are determined by the local spacing of float32 used to store edges. In Section 7, we probe the effect of minimum width by introducing jitter into the datasets.

4 Likelihood-maximizing intervals

The method proposed in this work is to fit a fixed piecewise-constant density function (no input history) by maximum likelihood and inherit the edges of this density as the edges used by the piecewise-constant output head. The fit involves both choosing the edges and the categorical weights, although only the edges are of interest thereafter. Such a fit will naturally place narrow intervals at concentrations of inter-event times.

For a fixed number of intervals, dynamic programming can obtain maximum-likelihood irregular histograms (Rozenholc et al., 2010). This is possible as the objective decomposes additively over intervals (Jackson et al., 2005). Related Bayesian construction of histograms by Simensen et al. (2026) and Scargle et al. (2013) also exploits this property, and it is the property we use here too.

4.1 FITTING A PIECEWISE-CONSTANT DENSITY FUNCTION TO INTER-EVENT TIMES

For a fixed N, we wish to find the set of edges $a_1, a_2, ..., a_N$ and weights m_i that maximize the data likelihood (of inter-event times) according to equation 2.

Let $\mathcal{D} = \{\tau_k\}_{k=1}^{K}$ be the inter-event times available for fitting, split into training and validation sets: $\mathcal{D}_{\text{train}} = \{\tau_k\}_{k=1}^{K_t}$ and $\mathcal{D}_{\text{val}} = \{\tau_k\}_{k=1}^{K_v}$. Denote s_i as the number of inter-event times from $\mathcal{D}_{\text{train}}$ that fall into the i-th interval, and s_i^* the same count for $\mathcal{D}_{\text{val}}$. To allow the exponential tail to be handled separately, the last edge is not fit, and instead is set to be the next representable value past the final sample, $a_N = \text{nextafter}(\max(\mathcal{D}))$, making the final interval empty of samples. The unconditional log-likelihood of $\mathcal{D}_{\text{val}}$ according to the N-interval piecewise density from equation 2 is the following sum over the first $N-1$ intervals:

$$\log p(\mathcal{D}_{\text{val}}) = \sum_{i=1}^{N-1} s_i^* (\log m_i - \log w_i), \qquad \text{where } w_i = a_{i+1} - a_i.$$

For fixed edges, the log-likelihood depends on mass through the term, $s_i^* \log m_i$. For the true data generating distribution, the expected log-likelihood is maximized by setting m_i to the fraction of inter-event times that would fall into the interval. We estimate this with the empirical frequencies from $\mathcal{D}_{\text{train}}$, setting m_i to s_i/K_t. With this substitution, the log-likelihood becomes:

$$\log p(\mathcal{D}_{\text{val}}) = \sum_{i=1}^{N-1} s_i^* (\log s_i - \log K_t - \log w_i)$$

To maximize this, we can ignore the constant term, $\log K_t$, and form the following objective function F for the choice of edges $\mathbf{a} = [a_1, a_2, \ldots, a_N]$:

$$F(\mathbf{a}) = \sum_{i=1}^{N-1} g(a_i, a_{i+1}), \quad \text{where } g(a_i, a_{i+1}) = s_i^* (\log s_i - \log w_i) \tag{3}$$

Each term of the sum can be calculated independently by knowing the edges of an interval and the number of events that fall within it. By maximizing F, we are estimating the piecewise-constant density function with the lowest cross-entropy to the data-generating distribution. We will maximize F over the possible choices of edges, $a_1, \cdots, a_N$.

4.2 OPTIMIZATION

While the objective to be maximized is simple, the search space—all possible sets of N edges—is vast. We restrict the edges to a finite set of candidates and, in doing so, make the problem amenable to dynamic programming.

Edge candidates. In this work, we create a set of candidates from three sources:

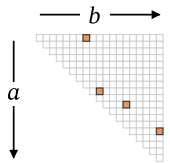
1. inter-event times from the data (but excluded from $\mathcal{D}$)
2. edges produced from the equal-quantile method (see Appendix B.1)
3. $e + \varepsilon$ for each candidate edge from 1) and 2) above, where ε is a small positive constant

By including the edges from the equal-quantile method, at least one set of edges can produce a likelihood similar to the equal-quantile method. The third source of candidates is included to allow intervals to have right edges just past candidate samples. For ε, we use the float32 unit in the last place (ULP) of the candidate edge.

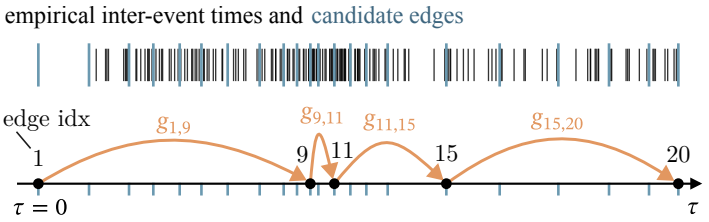
Dynamic programming. Consider a finite set of candidate edges $\mathcal{E} = \{e_1, e_2, \ldots, e_E\}$. The lowest and highest edges are $e_1 = 0$ and $e_E = a_N$. Note that typically $E \gg N$. Let $V(e, r)$ be the best value of F achievable with r edges left to place after edge e. In other words:

$$V(e, r) = \max_{e' \in \{x \in \mathcal{E} : x > e\}} \left[g(e, e') + V(e', r-1) \right]. \tag{4}$$

The optimal objective value is $V(e_1, N-1)$, with base case $V(e_E, 0) = 0$ and $V(e, 0) = -\infty$ for all $e < e_E$. A standard application of dynamic programming, described in Algorithm 1, gives the optimal edges.



(a) Score table: every candidate interval $[e_a, e_b)$ has an entry $g_{ab} = s^*_{ab}(\log s_{ab} - \log w_{ab})$, defined for $b > a$. Shaded cells are the chosen hops between edges.



(b) A path between 5 edges (the black nodes) selected from the candidate edges $\mathcal{E}$ (blue lines). Candidates are dense where the data is dense, allowing narrow intervals to (optionally) be chosen there.

Figure 5: Edge selection as a dynamic program. Choosing edges $\mathbf{a} = [a_1, \ldots, a_N]$ (with $a_1 = e_1 = 0$, $a_N = e_E$) to maximize $F(\mathbf{a}) = \sum_{i=1}^{N-1} g(a_i, a_{i+1})$ is equivalent to finding the highest-scoring path between the candidate edges $\mathcal{E} = \{e_1, \ldots, e_E\}$, where the score of interval $[e_a, e_b)$ is the table entry g_{ab} (a). A small candidate set is shown; in practice $\mathcal{E}$ is large.

Algorithm 1 Edge optimization. Dynamic programming optimization of edge placement to maximize data likelihood for unconditional piecewise-constant density.

Require: sorted candidate edges $e_1 = 0 < \cdots < e_E = a_N$; training set $\mathcal{D}_{\text{train}}$, validation set $\mathcal{D}_{\text{val}}$, number of intervals N.
Ensure: edges $a_1, \ldots, a_N$ maximize F over the candidate set
$s_{ab} \leftarrow$ count samples in interval $[e_a, e_b)$ from $\mathcal{D}_{\text{train}}$ {prefix sum}
$s^*_{ab} \leftarrow$ count samples in interval $[e_a, e_b)$ from $\mathcal{D}_{\text{val}}$ {prefix sum}
$g_{ab} \leftarrow s^*_{ab}\big(\log s_{ab} - \log(e_b - e_a)\big)$ for all $a < b$, and $g_{ab} \leftarrow 0$ where $s^*_{ab} = 0$ {create table of all interval scores}
$V[E, 0] \leftarrow 0$ and $V[a, 0] \leftarrow -\infty$ for all $a < E$ {base cases}
for $r = 1$ to $N - 1$ **do**
  for all candidate indices a **do**
    $V[a, r] \leftarrow \max_{b>a}\big(g_{ab} + V[b, r-1]\big)$
    $\text{choice}[a, r] \leftarrow$ the maximizing b {back-pointer}
  end for
end for
recover edges $a_1, \ldots, a_N$ by following choice from $(1, N-1)$
return edges $a_1, \ldots, a_N$

Complexity. To choose N edges from E candidate edges, the algorithm uses $O(E^2)$ time and memory to calculate and store the score matrix, g_{ab}. The bottom-up dynamic programming procedure filling in the V array uses $O(E^2 N)$ time and $O(EN)$ memory, making the overall complexity $O(E^2 N)$ time and $O(E^2 + EN)$ for memory. The $O(E^2 N)$ term dominates. In this work, we use $E = 15000$. In our implementation, vectorized array operations evaluate the E^2 transitions at each stage, substantially reducing the practical runtime (see Appendix F.1 for wall-clock runtimes).

Dataset splits. In this work, at most 10% of the dataset is sampled to source candidate edges. The remaining inter-event times are used to form $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{val}}$, which we implement as a 5-fold split. The inter-event times come entirely from the training set designated for the subsequent training of the neural network model.

The next section describes experiments that test this output representation.

5 Experiment Setup

We compare our proposed representation to existing output heads. All heads are paired with 3 model stems and trained on 14 real-world datasets. Below we describe the output heads, the model stems, the datasets and the training and evaluation procedure.

Table 1: Model stems used in experiments. See Appendix D for implementation details. Most heads from Section 5.1 were originally studied with recurrent neural network (RNN) stems.

Label	Stem description	Input	Layers	Heads	Embed dim.	Parameters
rnn	gated recurrent unit (Cho et al., 2014)	32	1	NA	64	13k
gpt-a	GPT-2 transformer (Radford et al., 2019)	128	2	4	16	108k
gpt-b	GPT-2 transformer	128	6	4	32	1.20M

5.1 Baseline output representations

Among the many diverse models designed for next event prediction, the preponderance of them use one of the following output representations. These output representations will be paired with the model stems from Table 1.

Mixture of log-normals (*logmix*). A network emits M tuples (μ_i, σ_i, π_i) which parameterize a mixture of M log-normal distributions. Investigated by Shchur et al. (2020) and Panos (2024). Unlike the approaches below, this represents a probability distribution directly, rather than as a hazard function. We use this head with $M = 64$ components.

Constant hazard (*const*). A network emits a scalar which parameterizes a constant hazard function (Huang et al., 2019; Li et al., 2018). A constant hazard corresponds to an exponential probability distribution.

Exponential hazard (*exp*). A network emits a scalar which parameterizes an exponential hazard function (Du et al., 2016; Upadhyay et al., 2018). An exponential hazard corresponds to a Gompertz distribution.

Softplus hazard (*softplus*). A network emits one scalar x which parameterizes a hazard function of the form $h(\tau) = \beta \operatorname{softplus}(\frac{\alpha\tau + x}{\beta})$. A softplus hazard does not have a corresponding analytical distributional form. A softplus hazard is used in multiple works (Mei & Eisner, 2017; Zuo et al., 2020; Zhang et al., 2020).

Neural network parameterized hazard (*nn*). A network takes in time as an input and evaluates the integral of a hazard function at that time. Investigated by Omi et al. (2019). This approach uses automatic differentiation to compute both the hazard and the corresponding density function.

Piecewise-continuous hazard. A network outputs an array of values which parameterizes a piecewise-continuous hazard function. This approach was investigated by Jing & Smola (2017), using equal-width intervals. We do not test the output head from Jing & Smola (2017); however, we do test a piecewise-continuous hazard on the same edges calculated in our proposed approach (Section 4) as an ablation in Section 7.

Example outputs from models using the *logmix* and *softplus* heads are shown in Figure 4. Limitations of existing heads are discussed in Appendix E.

5.2 Model stems

Each output head above is paired with 3 model stems listed in Table 1. The recurrent neural network (RNN) model is used so as to be similar to the recurrent networks used in the original papers introducing many of the heads. The other stems are transformer-based models in two sizes.

5.3 Datasets

All datasets used in this work are sequences of event times (or lists of such sequences). The datasets used were made available alongside Shchur et al. (2020), Lüdke et al. (2023) and Xue et al. (2024) and are described in further detail in Appendix C.

5.4 TRAINING AND EVALUATION

Models are trained to minimize NLL, and performance is evaluated in terms of NLL and mean absolute error (MAE). Models are trained 10 times for each dataset. The train/val/test split is randomly sampled for each repeat. Training and evaluation are described in more detail in Appendix F.

6 RESULTS

A subset of results is shown in Table 2, comparing *pc* and *logmix* heads using the RNN stem in terms of negative log-likelihood (NLL). Focus is given to the *logmix* head throughout this work, as it is the most effective among the existing approaches tested. Results for all stem and head combinations are reported in Table 5 and Table 6 in Appendix A. The *rnn-pc* model outperforms existing models on all datasets in terms of NLL, and has the lowest (best) mean rank across datasets in terms of MAE.

Table 2: *rnn-pc* and *rnn-logmix* evaluated on 14 real-world datasets in terms of test-set NLL and MAE. Both models share the same architecture up until the output representation. Values are means over 10 training repeats where each repeat uses a different random data split. Cells are coloured when the score difference, Δ, is statistically significant with respect to the standard errors (SE) of the means: $|\Delta| > 1.96\sqrt{\text{SE}^2_{\text{rnn-logmix}} + \text{SE}^2_{\text{rnn-pc}}}$; green when *rnn-pc* scores better, red when worse.

		NLL ↓		MAE ↓	
Dataset	Unit	rnn-logmix	rnn-pc	rnn-logmix	rnn-pc
Yelp airport	minutes	4.71	4.69	34.68	35.04
Yelp Mississauga	minutes	3.92	3.26	21.36	21.30
Twitter	minutes	4.04	3.02	37.23	37.34
Wikipedia	minutes	4.36	1.96	198.2	198.4
Yelp Toronto	hours	4.81	4.60	62.82	62.75
MOOC	minutes	1.02	-6.23	290.5	290.4
Amazon	days	1.44	1.38	13.58	13.41
Taobao	minutes	3.44	-2.01	21.30	21.31
PUBG	seconds	1.64	-5.87	18.32	17.51
Reddit Politics	seconds	4.68	0.29	53.89	53.72
Reddit AskScience	seconds	5.59	2.15	189.3	189.0
Last.fm	minutes	4.47	2.86	667.0	674.4
NYC taxi	minutes	4.15	-1.30	27.48	27.50
Stack Overflow	days	5.09	4.94	128.7	128.7

The next section describes a number of follow-up experiments.

7 ABLATIONS

We carry out five extensions to the experiment of the previous section.

Edge selection. We test the equal-width and equal-quantile methods for choosing edges from Section 3.1. Results are reported in Table 9 in Appendix B.1. In terms of mean rank across datasets, likelihood-maximizing edges beats equal-quantile which in turn beats equal-width, for both NLL and MAE. The equal-quantile model has a similar mean rank to the *rnn-logmix* model.

Edge count. We test a range of bin counts for the *pc* head, from 32 to 2048 and report results in Table 10 (NLL) and Table 11 (MAE) in Appendix B.2. For most datasets (but not all), NLL scores improve and MAE scores worsen as the number of bins increases. The degree to which NLL improves with increasing bin count correlates with the Gini coefficient of the datasets (see Figure 7).

Temporal precision. We perturb the event times with jitter in order to remove the reward for concentrating density in narrow intervals. Both the *pc* and *logmix* heads can concentrate density in narrow intervals (see Figure 4 for an example), and it is hypothesized that this is a factor contributing to their performance advantages over the other heads (Appendix E continues this argument).

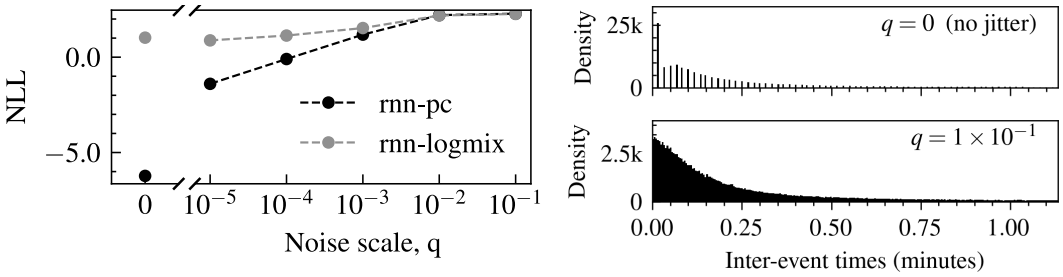


Figure 6: **Left:** NLL scores for *rnn-pc* and *rnn-logmix* on the MOOC dataset with jittered event times. Event times are jittered by adding Gaussian noise with standard deviation proportional to the median inter-event time, $\tilde{\tau}$. That is, $t_i^* = t_i + \eta$ where $\eta \sim \mathcal{N}(0, (\tilde{\tau}q)^2)$. Five values of q are used, from 1×10^{-5} to 1×10^{-1}. **Right:** histograms of the MOOC dataset's inter-event times for the two extreme values of the noise scale, $q = 0$ and $q = 1 \times 10^{-1}$.

Figure 6 shows the effect of jittering event times on the MOOC dataset. The same experiment is carried out for the other datasets, and the results appear in Figure 8 in Appendix B.3.

Density vs hazard. We test outputting a piecewise-constant hazard function with the same edges used by the *pc* head. A piecewise-constant hazard maps to a piecewise-exponential density. Results appear in Appendix B.4. The piecewise-constant hazard achieves broadly similar results. This suggests that the within-interval shape of the density function or whether it is formulated directly as a density or via a hazard function has less impact on performance than the selection of edge, and that our edge selection procedure can be effective for piecewise-continuous distributions with non-constant within-interval densities.

Synthetic dataset. We test the models on a sequence generated by simulating a Hawkes process. The simulation avoids the concentrations of density seen in many of the real-world datasets. Results appear in Appendix B.5. Models with a *logmix* head are the most performant in terms of NLL and MAE; the *pc* head comes second, and the *nn* head third.

8 Discussion

The results demonstrate that the piecewise-constant output head can be effective for next event prediction. To understand why this output structure outperforms existing heads, it is illustrative to inspect the datasets being used: many of the datasets have inter-event time distributions with shapes that are difficult for a model to approximate when using the existing output structures.

8.1 Discrete nature of events recorded from real-world processes

Recording inter-event times from a real-world process does not imply that the distribution of inter-event times will be continuous. Consider the sequence of taxi pickup times from the NYC taxi dataset (Whong, 2014) with inter-event time distribution shown in Figure 1. Inter-event times concentrate on minute boundaries, possibly due to the temporal resolution of the recording device or storage format. The discrete component in the distribution is not unique to the NYC taxi dataset. Inter-event time histograms for all the datasets used in this work are shown in Figures 9 to 12. Some of the histograms look smooth and some look clearly discrete. Some of the histograms that look smooth at one resolution have obvious discrete components when viewed at a finer resolution. The Stack Overflow badges dataset, which has an inter-event time distribution which appears smooth at multiple resolutions (see Figure 10f), has a strong concentration of density associated with yearly awarded badges which can be seen in the histograms of next event times in Figure 12.

8.2 Representational limitations of existing output heads

The nature of the datasets demand that models should be able to output distributions with multiple modes and with narrow peaks. All existing heads tested in this work have properties which limit

them with respect to these two goals. Appendix E covers the limitation of each head in detail. We summarize here: the constant, exponential and softplus hazard outputs correspond to unimodal distributions, and are also restricted in their ability to concentrate density at any non-zero time point. The mixture of lognormals head is far more flexible, but it encounters training instability in the presence of narrow peaks in the target distribution. Finally, the neural-network based head, while also capable of outputting complex distributions, is limited in its ability to modulate its output for different inputs.

In comparison, the piecewise-constant output is flexible enough to express distributions with discrete components and multiple modes, and it can be trained reliably to do so. We hypothesize that the proposed edge selection method allows for an effective allocation of predictive resolution. We further hypothesize that the within-interval shape (constant, exponential, etc.) is of secondary importance, especially as the interval count is increased. Some evidence to support this comes from the hazard function experiment from Section 7, where the piecewise-constant hazard output, which maps to a piecewise-exponential density function, performed similarly to the piecewise-constant density output structure.

8.3 Implications of the disappointing results of larger models

While increased flexibility of the output structure brought performance improvements, increased model capacity through larger parameter counts did not result in obvious improvements. All output heads were attached to three model stems: an RNN stem and two transformer models (see Table 1). The parameter counts spanned three orders of magnitude: 13k, 108k and 1.20M parameters. When scoring the stems in terms of mean rank NLL and MAE across datasets (Table 8), the largest model stem, *gpt-b*, is consistently the worst performing stem to be paired with any head. We draw on this result to hypothesize that larger models may perform less effectively on account of properties of the datasets. We hypothesize (and do not investigate) that qualities such as dataset size may mean that the regularization effects of model size give smaller models an advantage. If this were true, it would suggest that a benchmark formed of these commonly used datasets is insufficient to infer comparative model performance in contexts that demand larger models.

9 Limitations of this work

We argue that this work, and the field in general, is limited by the use of probability *density* for evaluating next event predictions. The distributions of inter-event times collected from real-world processes often contain sharp peaks, and it is suitable (and desirable) that models can represent these; however, evaluating in terms of density allows narrow intervals to concentrate density beyond what would be considered useful in a practical application. For example, if a taxi company wished to predict future pickup times, there would be a temporal resolution of interest, say minutes or seconds, beyond which there is little value in making predictions. These considerations make a probability mass function, rather than a density function, seem like an appropriate representation for making predictions. If there is no application to inform a minimum resolution of interest, then the resolution of the recording device used to capture the events could be used. We do not pursue this direction in this work. There would be further value if real-world *tasks* rather than simply event sequences from real-world processes were considered, as many of the current datasets used in this and prior work do not have an obvious connection to an application, and thus, they lack application-driven evaluation metrics.

10 Conclusion

For the task of next event prediction, this work proposes to have a model output a mostly piecewise-constant density function (or Appendix B.4's piecewise-constant hazard function). We present a dynamic programming optimization procedure for selecting the edges of this piecewise-constant density function. The proposed output representation achieves competitive performance. Where the approach markedly outperforms existing output representations, we hypothesize that our approach better represents the discrete components present in datasets' inter-event distributions. The results raise the question of whether probability density is always suitable for evaluating next event prediction models operating on event sequences recorded from real-world processes.

AI USE STATEMENT

In this work, we have used generative AI tools to: format LaTeX tables and to check for spelling, grammar and latex formatting errors. One or more authors had Copilot's autocomplete enabled while writing the supplementary code. The paper was submitted to a paper assistant tool made available through OpenReview which gave feedback on the paper. We believe that none of these uses constituted assistance in the categories we list below.

We have not used generative AI tools to: help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in the writing of proofs, propose or refine hypotheses, design research methodology or experiments, implement methods, assist with translation, clean and reformat datasets, support qualitative and thematic data analysis, or interpret results.

We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY

The supplementary material (code and datasets) permits all experiments to be reproduced. The Stack Overflow dataset is not included, as the usage agreement prevents distribution; for this case, instructions for obtaining the data are provided in Appendix C.3. To accompany the code, the appendix includes descriptions of models (Appendix D), datasets (Appendix C) and training procedures (Appendix F).

REFERENCES

Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006. ISBN 0-387-31073-8.

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*, March 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=3OSISBQPrM&referrer=%5BTMLR%5D(%2Fgroup%3Fid%3DTMLR).

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In Alessandro Moschitti, Bo Pang, and Walter Daelemans (eds.), *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A Meeting of SIGDAT, a Special Interest Group of the ACL*, pp. 1724–1734. ACL, 2014. doi: 10.3115/V1/D14-1179. URL https://doi.org/10.3115/v1/d14-1179.

Nan Du, Hanjun Dai, Rakshit Trivedi, Utkarsh Upadhyay, Manuel Gomez-Rodriguez, and Le Song. Recurrent Marked Temporal Point Processes: Embedding Event History to Vector. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pp. 1555–1564, New York, NY, USA, August 2016. Association for Computing Machinery. ISBN 978-1-4503-4232-2. doi: 10.1145/2939672.2939875. URL https://dl.acm.org/doi/10.1145/2939672.2939875.

Jeremy Howard and Sylvain Gugger. *Deep Learning for Coders with Fastai and PyTorch: AI Applications without a PhD*. O'Reilly Media, Inc, Sebastopol, California, first edition edition, 2020. ISBN 978-1-4920-4552-6.

Hengguan Huang, Hao Wang, and Brian Mak. Recurrent Poisson Process Unit for Speech Recognition. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):6538–6545, July 2019. ISSN 2374-3468. doi: 10.1609/aaai.v33i01.33016538. URL https://ojs.aaai.org/index.php/AAAI/article/view/4620.

Rob J. Hyndman and Yanan Fan. Sample Quantiles in Statistical Packages. *The American Statistician*, 50(4):361–365, 1996. ISSN 0003-1305. doi: 10.2307/2684934. URL https://www.jstor.org/stable/2684934.

B. Jackson, J.D. Scargle, D. Barnes, S. Arabhi, A. Alt, P. Gioumousis, E. Gwin, P. Sangtrakulcharoen, L. Tan, and Tun Tao Tsai. An algorithm for optimal partitioning of data on an interval. *IEEE Signal Processing Letters*, 12(2):105–108, February 2005. ISSN 1558-2361. doi: 10.1109/LSP.2001.838216. URL https://ieeexplore.ieee.org/document/1381461.

How Jing and Alexander J. Smola. Neural Survival Recommender. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, WSDM '17, pp. 515–524, New York, NY, USA, February 2017. Association for Computing Machinery. ISBN 978-1-4503-4675-7. doi: 10.1145/3018661.3018719. URL https://dl.acm.org/doi/10.1145/3018661.3018719.

Yuichiro Kanazawa. An Optimal Variable Cell Histogram Based on the Sample Spacings. *The Annals of Statistics*, 20(1):291–304, 1992. ISSN 0090-5364. URL https://www.jstor.org/stable/2242161.

Petri Kontkanen and Petri Myllymäki. MDL Histogram Density Estimation. In *Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics*, pp. 219–226. PMLR, March 2007. URL https://proceedings.mlr.press/v2/kontkanen07a.html.

Srijan Kumar, Xikun Zhang, and Jure Leskovec. Predicting Dynamic Embedding Trajectory in Temporal Interaction Networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, pp. 1269–1278, New York, NY, USA, July 2019. Association for Computing Machinery. ISBN 978-1-4503-6201-6. doi: 10.1145/3292500.3330895. URL https://dl.acm.org/doi/10.1145/3292500.3330895.

Håvard Kvamme and Ørnulf Borgan. Continuous and discrete-time survival prediction with neural networks. *Lifetime Data Analysis*, 27(4):710–736, October 2021. ISSN 1572-9249. doi: 10.1007/s10985-021-09532-6. URL https://doi.org/10.1007/s10985-021-09532-6.

Shuang Li, Shuai Xiao, Shixiang Zhu, Nan Du, Yao Xie, and Le Song. Learning Temporal Point Processes via Reinforcement Learning. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper/2018/hash/5d50d22735a7469266aab23fd8aeb536-Abstract.html.

Zichong Li, Qunzhi Xu, Zhenghao Xu, Yajun Mei, Tuo Zhao, and Hongyuan Zha. Beyond Point Prediction: Score Matching-based Pseudolikelihood Estimation of Neural Marked Spatio-Temporal Point Process. In *Forty-First International Conference on Machine Learning*, June 2024. URL https://openreview.net/forum?id=CpI37NA7MO&referrer=%5Bthe%20profile%20of%20Zichong%20Li%5D(%2Fprofile%3Fid%3D~Zichong_Li2).

Haitao Lin, Cheng Tan, Lirong Wu, Zicheng Liu, Zhangyang Gao, and Stan Z. Li. An Extensive Survey With Empirical Studies on Deep Temporal Point Process. *IEEE Transactions on Knowledge and Data Engineering*, 37(4):1599–1619, April 2025. ISSN 1558-2191. doi: 10.1109/TKDE.2024.3522114. URL https://ieeexplore.ieee.org/abstract/document/10815068.

Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

David Lüdke, Marin Biloš, Oleksandr Shchur, Marten Lienen, and Stephan Günnemann. Add and Thin: Diffusion for Temporal Point Processes. *Advances in Neural Information Processing Systems*, 36:56784–56801, December 2023. URL https://papers.neurips.cc/paper_files/paper/2023/hash/b1d9c7e7bd265d81aae8d74a7a6bd7f1-Abstract-Conference.html.

Hongyuan Mei and Jason Eisner. The neural hawkes process: A neurally self-modulating multivariate point process. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pp. 6757–6767, Red Hook, NY, USA, December 2017. Curran Associates Inc. ISBN 978-1-5108-6096-4.

Takahiro Omi, Naonori Ueda, and Kazuyuki Aihara. Fully Neural Network based Model for General Temporal Point Processes. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://papers.nips.cc/paper_files/paper/2019/hash/39e4973ba3321b80f37d9b55f63ed8b8-Abstract.html.

Aristeidis Panos. Decomposable Transformer Point Processes. *Advances in Neural Information Processing Systems*, 37:88932–88955, December 2024. doi: 10.52202/079017-2821. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/a1f0d96171c8e79019ae35ee439c2938-Abstract-Conference.html.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language Models are Unsupervised Multitask Learners. *Technical report, OpenAI*, February 2019.

Yves Rozenholc, Thoralf Mildenberger, and Ursula Gather. Combining regular and irregular histograms by penalized likelihood. *Computational Statistics & Data Analysis*, 54(12):3313–3323, December 2010. ISSN 0167-9473. doi: 10.1016/j.csda.2010.04.021. URL https://www.sciencedirect.com/science/article/pii/S0167947310001660.

Jeffrey D. Scargle, Jay P. Norris, Brad Jackson, and James Chiang. STUDIES IN ASTRONOMICAL TIME SERIES ANALYSIS. VI. BAYESIAN BLOCK REPRESENTATIONS. *The Astrophysical Journal*, 764(2):167, February 2013. ISSN 0004-637X. doi: 10.1088/0004-637X/764/2/167. URL https://doi.org/10.1088/0004-637X/764/2/167.

David W. Scott. Histogram. *WIREs Computational Statistics*, 2(1):44–48, 2010. ISSN 1939-0068. doi: 10.1002/wics.59. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/wics.59.

Oleksandr Shchur, Marin Biloš, and Stephan Günnemann. Intensity-Free Learning of Temporal Point Processes. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=HygOjhEYDH.

Oleksandr Shchur, Ali Caner Türkmen, Tim Januschowski, and Stephan Günnemann. Neural Temporal Point Processes: A Review. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, pp. 4585–4593, Montreal, Canada, August 2021. International Joint Conferences on Artificial Intelligence Organization. ISBN 978-0-9992411-9-6. doi: 10.24963/ijcai.2021/623. URL https://www.ijcai.org/proceedings/2021/623.

Oskar Høgberg Simensen, Dennis Christensen, and Nils Lid Hjort. Random irregular histograms. *Computational Statistics & Data Analysis*, 220:108367, August 2026. ISSN 0167-9473. doi: 10.1016/j.csda.2026.108367. URL https://www.sciencedirect.com/science/article/pii/S0167947326000368.

Leslie N. Smith. Cyclical Learning Rates for Training Neural Networks. *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 464–472, March 2017. doi: 10.1109/WACV.2017.58. URL http://ieeexplore.ieee.org/document/7926641/.

Leslie N. Smith and Nicholay Topin. Super-convergence: Very fast training of neural networks using large learning rates. In *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications*, volume 11006, pp. 369. SPIE, May 2019. doi: 10.1117/12.2520589. URL https://www.spiedigitallibrary.org/conference-proceedings-of-spie/11006/2520589/Super-convergence--very-fast-training-of-neural-networks-using/10.1117/12.2520589.

Stack Overflow. Stack Exchange Data Dump, August 2024. URL https://stackoverflow.com/help/data-dumps.

Utkarsh Upadhyay, Abir De, and Manuel Gomez Rodriguez. Deep Reinforcement Learning of Marked Temporal Point Processes. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper/2018/hash/71a58e8cb75904f24cde464161c3e766-Abstract.html.

Chris Whong. NYC's Taxi Trip Data, March 2014. URL https://chriswhong.com/open-data/foil_nyc_taxi/.

Siqiao Xue, Xiaoming Shi, James Zhang, and Hongyuan Mei. HYPRO: A Hybridly Normalized Probabilistic Model for Long-Horizon Prediction of Event Sequences. In *Advances in Neural Information Processing Systems*, volume 35, pp. 34641–34650. Curran Associates, Inc., 2022. doi: 10.52202/068431-2510. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/dfbb3d1807b21dadee735eb75069ada4-Abstract-Conference.html.

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 *The Twelfth International Conference on Learning Representations*, January 2024. URL https://openreview.net/forum?id=PJwAkg0z7h.

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

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

A SUPPLEMENTARY RESULTS

This appendix presents results supplementary to the main results in Section 6. The supplementary results are:

1. Appendix A.1: confidence intervals for the *rnn-logmix* vs *rnn-pc* table (Table 2).
2. Appendix A.2: an extension of Table 2 that includes score differences of all stem-head combinations, with *rnn-logmix* as a reference. These results also include two models implemented by following the papers by Zuo et al. (2020) and Panos (2024), *thp* and *dtpp* respectively.

For ablations, see Appendix B.

A.1 CONFIDENCE INTERVALS FOR MAIN RESULTS (TABLE 2)

In Section 6, Table 2 showed results for *rnn-logmix* and *rnn-pc* evaluated on 14 real-world datasets, averaged over 10 runs. These results are extended by Table 3 (NLL) and Table 4 (MAE) below to include information on the variability over the 10 runs. Training runs differ by the seed used to randomly split the datasets, by random model initialization and by the stochasticity of the training process.

Table 3: Confidence intervals for Table 2's NLL results (from Section 6). The *rnn-logmix* and *rnn-pc* models evaluated in terms of test set negative log-likelihood (NLL) (lower is better) on 14 real-world datasets. 10 training repeats were carried out. Datasets were randomly split on each repeat. Point estimates are the mean over the trials. Variability is expressed as 95% confidence intervals calculated assuming a normal distribution of the mean: $\pm 1.96 \frac{s}{\sqrt{10}}$, where s is the sample standard deviation.

Dataset	rnn-logmix	rnn-pc
Yelp airport	4.715 ± 2.16e-02	**4.685 ± 3.74e-02**
Yelp Mississauga	3.916 ± 1.93e-02	**3.260 ± 3.89e-02**
Twitter	4.043 ± 4.44e-02	**3.017 ± 7.35e-02**
Wikipedia	4.364 ± 6.35e-02	**1.962 ± 1.18e-01**
Yelp Toronto	4.810 ± 6.27e-02	**4.598 ± 6.98e-02**
MOOC	1.023 ± 1.38e-01	**-6.227 ± 2.90e-02**
Amazon	1.439 ± 8.81e-03	**1.380 ± 5.49e-03**
Taobao	3.440 ± 1.52e-02	**-2.010 ± 6.83e-02**
PUBG	1.643 ± 1.53e-01	**-5.872 ± 2.68e-02**
Reddit Politics	4.682 ± 4.48e-02	**0.288 ± 8.24e-02**
Reddit AskScience	5.585 ± 1.16e-01	**2.150 ± 9.73e-02**
Last.fm	4.465 ± 9.78e-02	**2.856 ± 2.26e-01**
NYC taxi	4.146 ± 6.21e-03	**-1.295 ± 4.27e-01**
Stack Overflow	5.091 ± 2.79e-02	**4.941 ± 2.55e-02**

A.2 ALL OUTPUT HEADS AND MODEL STEMS

In Table 5, the same 14 datasets are used to test all model combinations of the three stems (*rnn*, *gpt-a* and *gpt-b*) with the six heads: *const*, *exp*, *softplus*, *nn*, *logmix* and *pc*. Also included are two full models, *thp*, the model from *Transformer Hawkes Process* (Zuo et al., 2020), and *dtpp*, the model from *Decomposable Transformer Point Processes* (Panos, 2024). Table 6 shows the corresponding MAE scores.

These large tables are condensed by reporting mean ranks in Table 7 and Table 8.

Table 4: Confidence intervals for Table 2's MAE results (from Section 6). The *rnn-logmix* and *rnn-pc* models evaluated in terms of test set MAE (lower is better) on 14 real-world datasets. 10 training repeats were carried out. Datasets were randomly split on each repeat. Point estimates are the mean over the trials. Variability is expressed as 95% confidence intervals calculated assuming a normal distribution of the mean: $\pm 1.96 \frac{s}{\sqrt{10}}$, where s is the sample standard deviation.

Dataset	rnn-logmix	rnn-pc
Yelp airport	**34.68 ± 7.42e-01**	35.04 ± 7.75e-01
Yelp Mississauga	21.36 ± 4.65e-01	**21.30 ± 5.27e-01**
Twitter	**37.23 ± 6.05e-01**	37.34 ± 6.28e-01
Wikipedia	**198.2 ± 7.39e+00**	198.4 ± 7.44e+00
Yelp Toronto	62.82 ± 2.18e+00	**62.75 ± 2.19e+00**
MOOC	290.5 ± 4.07e+00	**290.4 ± 4.08e+00**
Amazon	13.58 ± 1.13e-01	**13.41 ± 9.73e-02**
Taobao	**21.30 ± 3.18e-01**	21.31 ± 3.17e-01
PUBG	18.32 ± 1.66e-01	**17.51 ± 6.83e-02**
Reddit Politics	53.89 ± 8.39e-01	**53.72 ± 8.30e-01**
Reddit AskScience	189.3 ± 4.83e+00	**189.0 ± 5.09e+00**
Last.fm	**667.0 ± 3.73e+01**	674.4 ± 3.79e+01
NYC taxi	**27.48 ± 9.12e-01**	27.50 ± 9.10e-01
Stack Overflow	**128.7 ± 1.25e+00**	128.7 ± 1.25e+00

Table 5: Extension of Table 2 in Section 6. Test set NLL scores (lower is better) for various models on 14 real-world datasets. 6 output heads (*logmix*, *pc*, *nn*, *softplus*, *exp* and *const*) are paired with 3 model stems (*rnn*, *gpt-a* and *gpt-b*). Additionally, *thp* from Zuo et al. (2020) and *dtpp* from Panos (2024) are included (see Appendix D.3). Point estimates are the means over 10 repeats, where each repeat uses a random data split. The *rnn-logmix* model is used as a baseline, and cells are coloured when the score difference to the *rnn-logmix* model, Δ, is statistically significant with respect to the standard errors (SE) of the means, $|\Delta| > 1.96\sqrt{\text{SE}^2_{logmix} + \text{SE}^2_{model}}$; green when better than *rnn-logmix*, red when worse. **Bold** entries are the lowest (best) score point estimates per dataset.

Model	Yelp a.	Yelp M.	Twitter	Taobao	Wiki.	Yelp Tor.	PUBG	MOOC	Reddit Ask.	Amazon	Reddit Pol.	Last.fm	NYC taxi	Stack Overflow
rnn-logmix	4.71	3.92	4.04	3.44	4.36	4.81	1.64	1.02	5.59	1.439	4.68	4.47	4.146	5.09
gpt-a-logmix	4.71	3.91	4.10	3.46	4.42	4.67	1.98	0.39	4.66	1.488	3.79	4.40	3.115	5.06
gpt-b-logmix	4.71	3.91	4.11	3.46	4.42	4.56	1.95	0.85	5.00	1.551	3.95	4.43	3.544	5.11
rnn-pc	4.69	**3.26**	**3.02**	**-2.01**	**1.96**	4.60	-5.87	**-6.23**	2.15	**1.380**	0.29	2.86	-1.295	4.94
gpt-a-pc	**4.67**	3.26	3.07	-1.99	2.02	4.54	**-5.89**	-6.20	**2.13**	1.385	**0.25**	**2.81**	**-1.915**	**4.94**
gpt-b-pc	4.68	3.27	3.08	-1.98	2.03	**4.44**	-5.89	-6.17	2.14	1.387	0.25	2.84	-1.913	4.94
rnn-nn	4.75	3.96	4.06	3.44	4.40	4.92	3.54	2.84	5.80	1.561	5.16	4.64	4.149	5.26
gpt-a-nn	4.76	3.94	4.12	3.46	4.44	4.95	3.09	2.58	5.80	1.494	5.17	4.51	4.155	5.25
gpt-b-nn	4.80	3.98	4.17	3.47	4.48	4.91	3.84	2.74	5.82	1.554	5.16	4.53	4.138	5.26
rnn-softplus	4.74	3.96	4.36	3.74	5.32	5.14	4.00	4.49	5.80	4.137	5.17	5.51	4.419	5.50
gpt-a-softplus	4.73	3.95	4.41	3.76	5.37	5.14	3.99	4.51	5.81	4.142	5.17	5.47	4.401	5.56
gpt-b-softplus	4.74	3.96	4.41	3.76	5.39	5.14	3.99	4.53	5.81	4.144	5.17	5.48	4.399	5.55
rnn-exp	4.73	3.96	4.35	3.74	5.34	5.14	4.00	5.12	5.80	4.133	5.17	5.70	4.421	5.50
gpt-a-exp	4.73	3.95	4.40	3.75	5.39	5.15	3.99	5.13	5.81	4.140	5.18	5.67	4.417	5.56
gpt-b-exp	4.74	3.96	4.40	3.76	5.41	5.14	3.99	5.14	5.81	4.137	5.17	5.65	4.401	5.55
rnn-const	4.75	3.99	4.58	4.09	5.90	5.15	4.01	6.61	5.82	4.364	5.17	6.05	4.573	5.56
gpt-a-const	4.75	3.98	4.62	4.12	6.00	5.16	4.00	6.53	5.82	4.366	5.17	6.01	4.524	5.56
gpt-b-const	4.75	3.98	4.62	4.12	6.02	5.16	4.00	6.53	5.82	4.365	5.17	6.04	4.523	5.56
thp-0	4.74	3.95	4.40	3.76	5.46	5.17	4.00	5.19	5.83	4.142	5.18	inf	inf	5.53
dtpp	4.74	3.92	4.09	3.50	4.39	4.94	3.57	1.88	5.80	1.557	5.13	4.51	4.139	5.16

Table 6: Extension of Table 2 in Section 6. Test set MAE scores (lower is better) for various models on 14 real-world datasets. 6 output heads (*logmix*, *pc*, *nn*, *softplus*, *exp* and *const*) are paired with 3 model stems (*rnn*, *gpt-a* and *gpt-b*). Additionally, *thp* from Zuo et al. (2020) and *dtpp* from Panos (2024) are included (see Appendix D.3). Point estimates are the means over 10 repeats, where each repeat uses a random data split. The *rnn-logmix* model is used as a baseline, and cells are coloured when the score difference to the *rnn-logmix* model, Δ, is statistically significant with respect to the standard errors (SE) of the means, $|\Delta| > 1.96\sqrt{\mathrm{SE}^2_{\text{logmix}} + \mathrm{SE}^2_{\text{model}}}$; green when better than *rnn-logmix*, red when worse. **Bold** entries are the lowest (best) score point estimates per dataset.

Model	Yelp a.	Yelp M.	Twitter	Taobao	Wiki.	Yelp Tor.	PUBG	MOOC	Reddit Ask.	Amazon	Reddit Pol.	Last.fm	NYC taxi	Stack Overflow
rnn-logmix	34.68	21.36	**37.23**	21.30	**198.2**	62.82	18.32	290.5	189.3	13.58	53.89	**667.0**	27.48	128.7
gpt-a-logmix	34.55	21.42	37.96	21.41	199.8	62.94	17.49	290.5	191.1	13.46	54.04	670.2	27.57	**128.6**
gpt-b-logmix	**34.44**	21.43	37.98	21.43	200.1	62.86	17.49	290.5	189.3	13.47	53.77	672.0	27.48	128.9
rnn-pc	35.04	**21.30**	37.34	21.31	198.4	**62.75**	17.51	**290.4**	189.0	**13.41**	53.72	674.4	27.50	128.7
gpt-a-pc	34.70	21.55	38.07	21.46	199.9	63.02	17.46	290.5	190.0	13.47	53.63	673.6	**27.40**	128.8
gpt-b-pc	34.59	21.43	38.05	21.46	200.2	62.85	17.44	290.5	190.4	13.46	53.64	673.7	27.43	129.0
rnn-nn	35.06	21.60	37.32	**21.29**	198.5	63.12	17.59	291.3	**188.9**	13.99	53.75	674.1	27.49	129.1
gpt-a-nn	34.70	21.44	37.88	21.38	200.0	63.58	17.49	290.8	191.2	13.98	53.92	680.6	27.62	129.8
gpt-b-nn	34.99	21.63	37.79	21.39	200.3	63.10	17.50	290.9	193.1	13.99	53.70	680.6	27.51	129.2
rnn-softplus	35.00	21.51	38.28	21.74	204.1	62.90	17.54	291.9	189.1	16.58	53.69	722.8	27.71	131.9
gpt-a-softplus	34.72	21.45	38.77	21.88	205.2	63.02	17.46	291.1	190.3	16.49	**53.55**	695.0	27.86	132.3
gpt-b-softplus	34.75	21.53	38.65	21.93	205.4	63.05	17.46	291.1	190.3	16.55	53.58	694.5	27.91	132.2
rnn-exp	34.76	21.48	38.17	21.70	205.3	62.90	17.53	302.4	189.2	16.54	53.70	701.3	27.72	131.9
gpt-a-exp	34.66	21.45	38.55	21.87	205.6	63.45	**17.44**	301.4	190.0	16.60	54.03	699.2	27.88	133.0
gpt-b-exp	34.77	21.44	38.51	21.90	207.0	63.06	17.45	301.9	190.2	16.34	53.58	698.5	27.90	132.1
rnn-const	35.39	21.94	42.27	24.71	248.3	63.03	17.64	448.3	192.1	20.28	53.72	781.3	30.32	136.2
gpt-a-const	35.02	21.94	43.38	24.99	254.2	63.32	17.56	450.9	192.9	20.39	53.60	785.7	30.81	136.7
gpt-b-const	34.99	22.00	43.12	25.23	250.6	63.32	17.56	458.5	193.4	20.44	53.62	810.1	30.91	137.0
thp-0	45.56	26.28	41.07	23.10	202.6	79.93	22.94	290.6	245.7	28.45	75.91	708.0	38.60	153.4
dtpp	34.73	21.36	37.77	21.35	199.3	63.00	17.52	290.5	189.6	13.44	53.68	671.9	27.47	129.1

Table 7: Mean rank of models across all 14 datasets, in terms of NLL and MAE. The scores are taken from Table 5 and Table 6. **Bold** entries are the lowest (best) mean rank for each metric (i.e. per column).

	NLL rank ↓		MAE rank ↓	
Model	#	mean	#	mean
rnn-logmix	5	5.36	2	5.29
gpt-a-logmix	4	4.86	7	6.71
gpt-b-logmix	6	5.71	5	6.29
rnn-pc	2	2.14	1	**5.14**
gpt-a-pc	1	**1.50**	6	6.64
gpt-b-pc	3	2.43	3	5.79
rnn-nn	8	9.07	8	9.21
gpt-a-nn	9	9.14	9	9.64
gpt-b-nn	10	10.64	10	10.64
rnn-softplus	11	12.79	14	11.71
gpt-a-softplus	13	12.93	11	10.79
gpt-b-softplus	14	13.86	15	11.86
rnn-exp	11	12.79	12	11.29
gpt-a-exp	16	14.21	16	12.64
gpt-b-exp	14	13.86	13	11.57
rnn-const	18	18.07	17	16.86
gpt-a-const	18	18.07	18	16.93
gpt-b-const	20	18.43	19	17.43
thp-0	17	16.29	20	17.79
dtpp	7	7.86	3	5.79

Table 8: Mean rank of three model stems—*rnn*, *gpt-a* and *gpt-b*—in terms of NLL and MAE across the 14 real-world datasets. In addition to this rank (which appears in the final row), the mean rank is also calculated (for the same datasets) by first aggregating by output head. Each row has an NLL and a MAE ranking. The scores are taken from Table 5 and Table 6. **Bold** entries are the lowest (best) mean rank across the three model stems (i.e. per row, but separate for the two metrics).

	NLL rank ↓			MAE rank ↓		
Head	rnn	gpt-a	gpt-b	rnn	gpt-a	gpt-b
logmix	2.21	**1.50**	2.29	**1.71**	2.07	2.21
pc	2.07	**1.50**	2.43	**1.71**	2.07	2.21
nn	1.93	**1.79**	2.29	**1.79**	2.00	2.21
softplus	**1.79**	**1.79**	2.43	1.93	**1.86**	2.21
exp	**1.64**	2.21	2.14	**1.79**	2.14	2.07
const	**1.93**	**1.93**	2.14	**1.50**	2.00	2.50
all heads	1.93	**1.79**	2.29	**1.74**	2.02	2.24

Table 9: With *rnn-logmix* as a reference, the three edge selection methods are compared: *rnn-wpc* (equal-width), *rnn-qpc* (equal-quantile) and *rnn-pc* (maximum likelihood). All models share the same architecture up until the output representation. They are evaluated on 14 datasets in terms of test-set NLL and MAE. Values are means over 10 training repeats, where each repeat uses a random data split. Cells are coloured when the score difference to the *rnn-logmix* model, Δ, is statistically significant with respect to the standard errors (SE) of the means, $|\Delta| > 1.96\sqrt{\text{SE}^2_{\text{rnn-logmix}} + \text{SE}^2_{\text{model}}}$; green when a model is better than *rnn-logmix*, red when worse.

	NLL ↓				MAE ↓			
Dataset	rnn-logmix	rnn-pc	rnn-qpc	rnn-wpc	rnn-logmix	rnn-pc	rnn-qpc	rnn-wpc
Yelp airport	4.715	4.685	4.779	4.772	34.68	35.04	35.11	35.10
Yelp Miss.	3.916	3.260	3.949	3.968	21.36	21.30	21.64	21.52
Twitter	4.043	3.017	4.089	4.173	37.23	37.34	37.54	37.50
Wiki.	4.364	1.962	4.385	5.417	198.2	198.4	198.5	211.6
Yelp Toronto	4.810	4.598	4.808	5.005	62.82	62.75	62.75	62.79
MOOC	1.023	-6.227	-4.189	5.470	290.5	290.4	290.4	341.9
Amazon	1.439	1.380	1.447	1.464	13.58	13.41	13.41	13.38
Taobao	3.440	-2.010	2.493	3.494	21.30	21.31	21.33	21.34
PUBG	1.643	-5.872	-3.021	3.922	18.32	17.51	17.51	17.52
Reddit Politics	4.682	0.288	3.806	5.155	53.89	53.72	53.70	53.71
Reddit AskSci.	5.585	2.150	5.486	5.851	189.3	189.0	188.6	189.9
Last.fm	4.465	2.856	4.484	6.571	667.0	674.4	667.7	793.8
NYC taxi	4.146	-1.295	0.433	4.199	27.48	27.50	27.50	27.49
Stack Overflow	5.091	4.941	5.119	5.405	128.7	128.7	128.7	129.0

B Ablations

Five ablations were carried out:

1. Appendix B.1: varying the method used to select edges.
2. Appendix B.2: varying the number of edges.
3. Appendix B.4: switching from a piecewise-constant density function to a piecewise-constant hazard function.
4. Appendix B.3: perturbing event times to investigate the effect of narrow intervals.
5. Appendix B.5: using event sequences from a synthetic process (rather than recorded from real-world processes).

B.1 Edge selection method

In addition to the main approach of this work, Section 3.1 also mentioned two other ways to select edges to form a piecewise distribution: equal-width and equal-quantile. In this section, we compare all three edge selection approaches.

B.1.1 Equal-width and equal-quantile intervals. Details.

We implement the equal-width and equal-quantile edge selection methods as follows. For equal-width edge selection, the $N-2$ central intervals split the region between a_2 and a_N evenly by width. For equal-quantile edge selection, those same central intervals are split evenly by quantile. Quantiles are estimated from the training set following the method recommended by Hyndman & Fan (1996). In both equal-width and equal-quantile approaches, the first and last intervals are assigned length by quantile: they are given small probability masses m_1 and m_N (we use $m_1 = m_N = \frac{1}{4N}$) so they can cover the potentially long tails on either side of a distribution without being responsible for as many events as the central intervals.

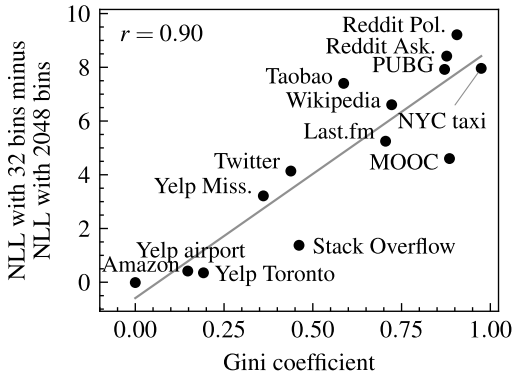


Figure 7: Gini coefficient for each dataset plotted against the NLL score difference obtained by two *rnn-pc* model variants: using 32 output bins and using 2048 output bins. Datasets with a higher Gini coefficient tend to benefit from the *rnn-pc* model using more output bins (higher NLL difference). This trend suggests that the Gini coefficient (and possibly other measures of distribution inequality) can be used to guide the choice of bin count for the *rnn-pc* model.

B.1.2 Results

Table 9 reports NLL and MAE results for the three different edge selection methods:

1. equal-width intervals (*wpc*)
2. equal-quantile intervals (*qpc*)
3. intervals which maximize likelihood for an unconditional model (*pc*) (ours)

All models are tested with the *rnn* stem and using 128 output intervals. The results for *rnn-logmix* are included for comparison. It is noteworthy that the piecewise-continuous density with equal-quantile edges also outperforms the mixture of lognormals (in terms of mean rank across datasets for both NLL and MAE).

B.2 Edge count

The *rnn-pc* model is tested over a range of output bin counts from 32 to 2048, with results appearing in Table 10a (NLL) and Table 11a (MAE). The results are reported as means over 10 runs, where a run involves training and evaluating a model with a random data split. There is no bin count that is best across all datasets. A possible explanation for the heterogeneous results is that expressive power and susceptibility to overfitting are both increased with increasing bin count. In Table 10b and Table 11b, the same results are shown with confidence intervals to indicate variability over the 10 runs.

B.3 Temporal precision

In this section, we perturb the event times with jitter in order to remove the reward for concentrating density in narrow intervals. Figure 8 shows the effect of jittering event times across all 14 datasets on the NLL scores for *rnn-pc* and *rnn-logmix*. Event times are jittered by adding Gaussian noise with standard deviation proportional to the median inter-event time, $\tilde{\tau}$. That is, $t_i^* = t_i + \eta$ where $\eta \sim \mathcal{N}(0, (\tilde{\tau}q)^2)$. Five values of q are used, from 1×10^{-5} to 1×10^{-1}. Both models' scores worsen under increased perturbation.

The *rnn-pc* model's score continues to improve even when removing the lowest level of noise. This is evidence to support the hypothesis that the *rnn-pc* model is rewarded for concentrating density in narrow intervals. For multiple datasets, such as the NYC taxi dataset (Figure 8m), the *rnn-pc* begins to outperform the *rnn-logmix* even when moderate ($q = 10^{-3}$) noise is present. This suggests that the *rnn-pc* does not rely on arbitrarily narrow intervals (down to float32 ULP) to outperform the *rnn-logmix* model. For most datasets, the *rnn-logmix* model's score improves to a plateau as noise is reduced.

Table 10: The *rnn-pc* model with various output bin counts, evaluated in terms of test set NLL (lower is better) on 14 real-world datasets. Values are means over 10 repeats. Table (a) reports point estimates with the 128-bin model as a reference, and Table (b) reports point estimates with confidence intervals.

(a) With the 128-bin model (*rnn-pc*) as a reference, the other bin count models are compared in terms of test set NLL. Values are means over 10 training repeats, where each repeat uses a random data split. Cells are coloured when the score difference to the *rnn-pc* model, Δ, is statistically significant with respect to the standard errors (SE) of the means, $|\Delta| > 1.96\sqrt{\text{SE}^2_{\text{rnn-pc}} + \text{SE}^2_{\text{model}}}$; green when a model is better than *rnn-pc*, red when worse.

Model	Yelp a.	Yelp M.	Twitter	Taobao	Wiki.	Yelp Tor.	PUBG	MOOC	Reddit Ask.	Amazon	Reddit Pol.	Last.fm	NYC taxi	Stack Overflow
rnn-pc	4.69	3.26	3.02	-2.01	1.96	4.60	-5.87	-6.23	2.15	1.380	0.29	2.86	-1.295	4.94
rnn-pc-32	4.71	3.72	3.84	1.52	3.90	4.82	-1.33	-3.94	4.59	**1.378**	3.41	4.09	1.168	5.05
rnn-pc-64	**4.67**	3.53	3.50	-0.12	3.15	4.72	-3.77	-5.35	3.55	1.378	2.06	3.61	-0.548	4.98
rnn-pc-256	4.70	2.92	2.36	-3.66	0.48	4.51	-7.59	-6.86	0.45	1.382	-1.70	1.74	-1.593	4.92
rnn-pc-512	4.73	2.50	1.63	-4.84	-0.96	4.45	-8.61	-7.45	-1.30	1.385	-3.44	0.51	-2.046	**4.90**
rnn-pc-1024	4.80	2.32	1.11	-5.42	-2.15	**4.41**	-8.95	-8.15	-2.77	1.390	-4.68	-0.92	-2.918	4.90
rnn-pc-2048	4.86	**2.26**	**1.03**	**-5.47**	**-2.91**	4.42	**-8.97**	**-8.92**	**-3.78**	1.394	**-5.35**	**-2.10**	**-4.227**	4.92

(b) Confidence intervals for Table (a) above. 10 training repeats were carried out. Point estimates are the means over the 10 repeats. Variability is expressed as 95% confidence intervals calculated assuming a normal distribution of the mean: $\pm 1.96\frac{s}{\sqrt{10}}$, where s is the sample standard deviation.

Dataset	rnn-pc	rnn-pc-32	rnn-pc-64	rnn-pc-256	rnn-pc-512	rnn-pc-1024	rnn-pc-2048
Yelp a.	4.685 ± 3.74e-02	4.712 ± 2.88e-02	**4.674 ± 3.12e-02**	4.703 ± 3.11e-02	4.731 ± 2.97e-02	4.796 ± 3.33e-02	4.865 ± 4.63e-02
Yelp M.	3.260 ± 3.89e-02	3.722 ± 2.91e-02	3.528 ± 4.38e-02	2.918 ± 6.16e-02	2.503 ± 7.14e-02	2.318 ± 8.99e-02	**2.257 ± 8.04e-02**
Twitter	3.017 ± 7.35e-02	3.836 ± 5.04e-02	3.497 ± 6.90e-02	2.363 ± 1.03e-01	1.631 ± 1.13e-01	1.105 ± 1.14e-01	**1.031 ± 1.16e-01**
Wiki.	1.962 ± 1.18e-01	3.898 ± 7.33e-02	3.148 ± 9.33e-02	0.485 ± 1.48e-01	-0.961 ± 1.71e-01	-2.149 ± 1.79e-01	**-2.908 ± 1.82e-01**
Yelp Tor.	4.598 ± 6.98e-02	4.819 ± 6.44e-02	4.715 ± 6.93e-02	4.509 ± 7.18e-02	4.446 ± 7.70e-02	**4.408 ± 8.29e-02**	4.421 ± 7.71e-02
MOOC	-6.227 ± 2.90e-02	-3.940 ± 3.23e-02	-5.346 ± 2.90e-02	-6.862 ± 2.59e-02	-7.448 ± 2.44e-02	-8.152 ± 2.28e-02	**-8.915 ± 1.97e-02**
Amazon	1.380 ± 5.49e-03	**1.378 ± 5.55e-03**	1.378 ± 5.29e-03	1.382 ± 5.60e-03	1.385 ± 5.54e-03	1.390 ± 5.80e-03	1.394 ± 5.69e-03
Taobao	-2.010 ± 6.83e-02	1.519 ± 3.59e-02	-0.124 ± 5.36e-02	-3.662 ± 8.74e-02	-4.839 ± 8.68e-02	-5.425 ± 8.79e-02	**-5.472 ± 8.80e-02**
PUBG	-5.872 ± 2.68e-02	-1.333 ± 2.25e-02	-3.771 ± 2.57e-02	-7.585 ± 2.83e-02	-8.609 ± 3.17e-02	-8.947 ± 3.67e-02	**-8.969 ± 3.47e-02**
Reddit Pol.	0.288 ± 8.24e-02	3.413 ± 5.12e-02	2.063 ± 6.91e-02	-1.697 ± 8.47e-02	-3.441 ± 7.70e-02	-4.680 ± 6.84e-02	**-5.348 ± 6.64e-02**
Reddit Ask.	2.150 ± 9.73e-02	4.586 ± 5.79e-02	3.549 ± 8.10e-02	0.445 ± 1.01e-01	-1.296 ± 1.04e-01	-2.770 ± 9.58e-02	**-3.781 ± 8.39e-02**
Last.fm	2.856 ± 2.26e-01	4.093 ± 1.73e-01	3.605 ± 2.09e-01	1.744 ± 2.49e-01	0.507 ± 2.60e-01	-0.921 ± 2.62e-01	**-2.097 ± 2.47e-01**
NYC taxi	-1.295 ± 4.27e-01	1.168 ± 2.49e-01	-0.548 ± 3.69e-01	-1.593 ± 4.35e-01	-2.046 ± 4.04e-01	-2.918 ± 3.46e-01	**-4.227 ± 2.60e-01**
Stack Overflow	4.941 ± 2.55e-02	5.050 ± 2.74e-02	4.983 ± 2.97e-02	4.917 ± 3.00e-02	**4.903 ± 3.20e-02**	4.904 ± 2.86e-02	4.923 ± 2.84e-02

Table 11: The *rnn-pc* model with various output bin counts, evaluated in terms of test set MAE (lower is better) on 14 real-world datasets. Values are means over 10 repeats. Table (a) reports point estimates with the 128-bin model as a reference, and Table (b) reports point estimates with confidence intervals.

(a) With the 128-bin model (*rnn-pc*) as a reference, the other bin count models are compared in terms of test set MAE. Values are means over 10 training repeats, where each repeat uses a random data split. Cells are coloured when the score difference to the *rnn-pc* model, Δ, is statistically significant with respect to the standard errors (SE) of the means, $|\Delta| > 1.96\sqrt{\mathrm{SE}^2_{\text{rnn-pc}} + \mathrm{SE}^2_{\text{model}}}$; green when a model is better than *rnn-pc*, red when worse.

Model	Yelp a.	Yelp M.	Twitter	Taobao	Wiki.	Yelp Tor.	PUBG	MOOC	Reddit Ask.	Amazon	Reddit Pol.	Last.fm	NYC taxi	Stack Overflow
rnn-pc	35.04	**21.30**	37.34	**21.31**	198.4	62.75	17.51	290.4	189.0	13.41	53.72	674.4	27.50	128.7
rnn-pc-32	**34.70**	21.36	**37.31**	21.42	198.5	**62.75**	17.64	290.4	190.2	**13.38**	53.78	681.1	27.55	**128.7**
rnn-pc-64	34.79	21.36	37.32	21.33	**198.3**	62.76	**17.51**	**290.4**	189.2	13.39	53.75	686.1	**27.50**	128.7
rnn-pc-256	35.38	21.39	37.37	21.32	198.4	62.75	17.51	290.4	188.7	13.43	53.71	668.8	27.50	128.7
rnn-pc-512	35.38	21.57	37.39	21.34	198.4	62.76	17.52	290.4	**188.7**	13.45	**53.71**	668.4	27.50	128.7
rnn-pc-1024	35.38	22.05	37.48	21.36	198.5	62.82	17.52	290.5	188.8	13.47	53.72	**668.2**	27.51	128.8
rnn-pc-2048	35.38	22.76	38.92	21.53	198.6	62.92	17.52	290.5	188.9	13.54	53.77	668.4	27.52	129.0

(b) Confidence intervals for Table (a) above. 10 training repeats were carried out. Point estimates are the means over the 10 repeats. Variability is expressed as 95% confidence intervals calculated assuming a normal distribution of the mean: $\pm 1.96\frac{s}{\sqrt{10}}$, where s is the sample standard deviation.

Dataset	rnn-pc	rnn-pc-32	rnn-pc-64	rnn-pc-256	rnn-pc-512	rnn-pc-1024	rnn-pc-2048
Yelp a.	35.04 ± 7.75e-01	**34.70 ± 8.23e-01**	34.79 ± 8.00e-01	35.38 ± 7.42e-01	35.38 ± 7.43e-01	35.38 ± 7.40e-01	35.38 ± 7.49e-01
Yelp M.	**21.30 ± 5.27e-01**	21.36 ± 5.21e-01	21.36 ± 5.71e-01	21.39 ± 5.99e-01	21.57 ± 5.76e-01	22.05 ± 5.69e-01	22.76 ± 5.85e-01
Twitter	37.34 ± 6.28e-01	**37.31 ± 6.21e-01**	37.32 ± 6.19e-01	37.37 ± 6.29e-01	37.39 ± 6.32e-01	37.48 ± 6.33e-01	38.92 ± 6.28e-01
Wiki.	198.4 ± 7.44e+00	198.5 ± 7.37e+00	**198.3 ± 7.39e+00**	198.4 ± 7.42e+00	198.4 ± 7.39e+00	198.5 ± 7.42e+00	198.6 ± 7.41e+00
Yelp Tor.	62.75 ± 2.19e+00	**62.75 ± 2.19e+00**	62.76 ± 2.19e+00	62.75 ± 2.17e+00	62.76 ± 2.17e+00	62.82 ± 2.17e+00	62.92 ± 2.16e+00
MOOC	290.4 ± 4.08e+00	290.4 ± 4.07e+00	**290.4 ± 4.09e+00**	290.4 ± 4.08e+00	290.4 ± 4.08e+00	290.5 ± 4.07e+00	290.5 ± 4.07e+00
Amazon	13.41 ± 9.73e-02	**13.38 ± 9.60e-02**	13.39 ± 1.03e-01	13.43 ± 1.08e-01	13.45 ± 1.08e-01	13.47 ± 1.05e-01	13.54 ± 1.14e-01
Taobao	**21.31 ± 3.17e-01**	21.42 ± 3.17e-01	21.33 ± 3.14e-01	21.32 ± 3.17e-01	21.34 ± 3.14e-01	21.36 ± 3.20e-01	21.53 ± 3.19e-01
PUBG	17.51 ± 6.83e-02	17.64 ± 9.68e-02	**17.51 ± 6.77e-02**	17.51 ± 7.02e-02	17.52 ± 6.97e-02	17.52 ± 6.84e-02	17.52 ± 7.06e-02
Reddit Pol.	53.72 ± 8.30e-01	53.78 ± 8.30e-01	53.75 ± 8.34e-01	53.71 ± 8.36e-01	**53.71 ± 8.36e-01**	53.72 ± 8.37e-01	53.77 ± 8.37e-01
Reddit Ask.	189.0 ± 5.09e+00	190.2 ± 4.84e+00	189.2 ± 5.24e+00	188.7 ± 4.92e+00	**188.7 ± 4.88e+00**	188.8 ± 4.89e+00	188.9 ± 4.90e+00
Last.fm	674.4 ± 3.79e+01	681.1 ± 4.09e+01	686.1 ± 4.03e+01	668.8 ± 3.61e+01	668.4 ± 3.71e+01	**668.2 ± 3.67e+01**	668.4 ± 3.73e+01
NYC taxi	27.50 ± 9.10e-01	27.55 ± 9.11e-01	**27.50 ± 9.11e-01**	27.50 ± 9.12e-01	27.50 ± 9.10e-01	27.51 ± 9.09e-01	27.52 ± 9.10e-01
Stack Overflow	128.7 ± 1.25e+00	**128.7 ± 1.25e+00**	128.7 ± 1.26e+00	128.7 ± 1.25e+00	128.7 ± 1.26e+00	128.8 ± 1.25e+00	129.0 ± 1.28e+00

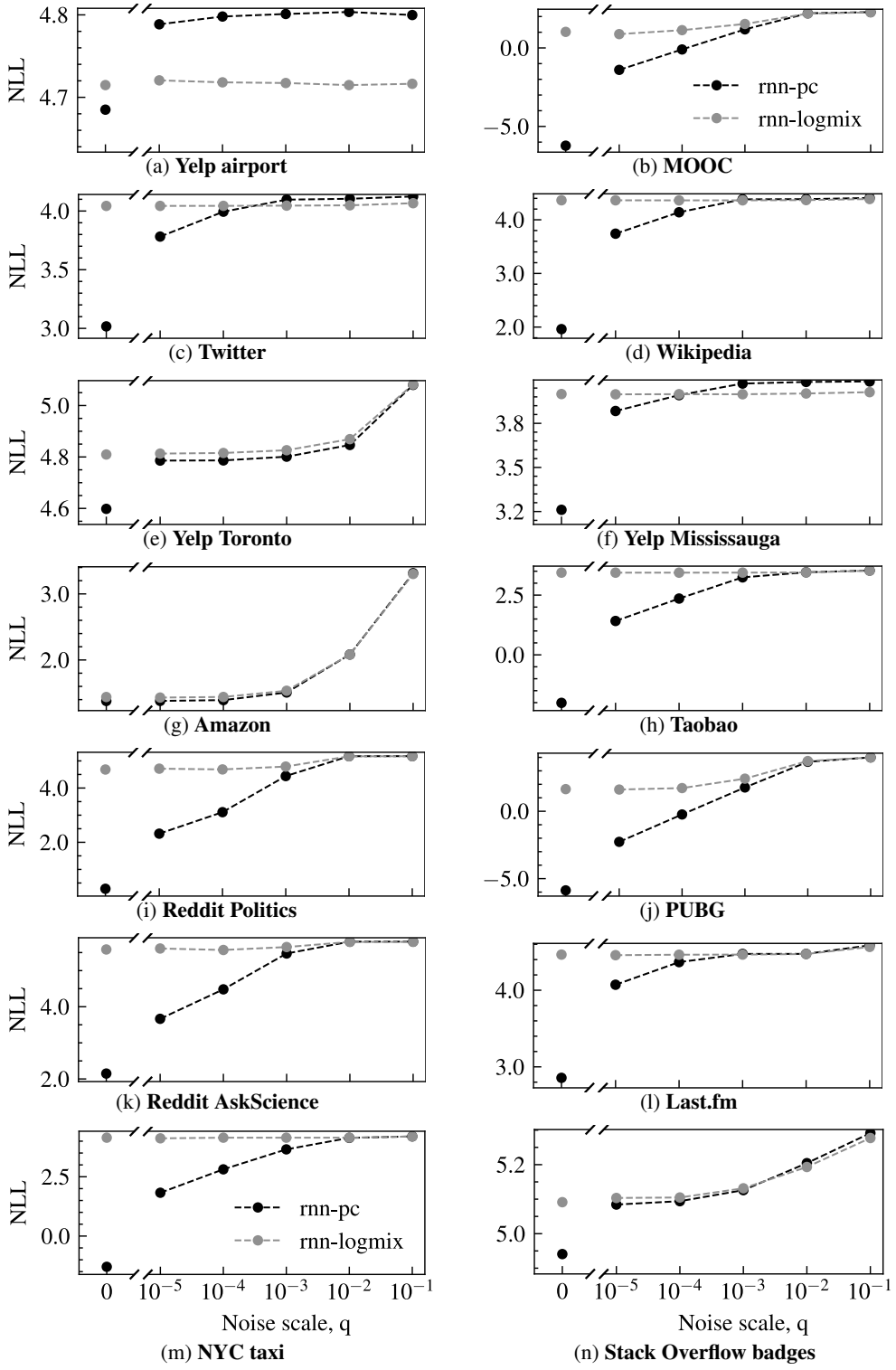


Figure 8: NLL scores for *rnn-pc* and *rnn-logmix* on 14 datasets with jittered event times. Event times are jittered by adding Gaussian noise with standard deviation proportional to the median inter-event time, $\tilde{\tau}$. That is, $t_i^* = t_i + \eta$ where $\eta \sim \mathcal{N}(0, (\tilde{\tau}q)^2)$. Five values of q are used, from 1×10^{-5} to 1×10^{-1}. Values are means over 10 repeats, where each repeat involves a new random data split and noise seed. Both models' scores improve under lower noise, but more so for *rnn-pc*.

Table 12: Density versus hazard. A piecewise-constant hazard output paired with the *rnn* model is compared to the *rnn-pc* model. Two piecewise-constant hazard outputs are considered: *rnn-qpch* and *rnn-pch*. They differ in the edge choice method—*rnn-qpch* uses the equal-quantile method and *rnn-pch* uses the maximum likelihood method. The models are evaluated on the 14 real-world datasets in terms of test-set NLL and MAE. All models share the same architecture up until the output representation. Values are means over 10 training repeats, where each repeat uses a random data split. Cells are coloured when the score difference to the *rnn-pc* model, Δ, is statistically significant with respect to the standard errors (SE) of the means, $|\Delta| > 1.96\sqrt{\mathrm{SE}^2_{\text{rnn-pc}} + \mathrm{SE}^2_{\text{rnn-}x}}$; green when a model is better than *rnn-pc*, red when worse.

	NLL ↓			MAE ↓		
Dataset	rnn-pc	rnn-qpch	rnn-pch	rnn-pc	rnn-qpch	rnn-pch
Yelp airport	4.685	4.781	4.682	35.04	35.22	35.16
Yelp Mississauga	3.260	3.947	3.234	21.30	21.78	21.61
Twitter	3.017	4.089	2.998	37.34	37.55	37.37
Wikipedia	1.962	4.376	1.927	198.4	198.7	198.4
Yelp Toronto	4.598	4.805	4.646	62.75	62.76	62.77
MOOC	-6.227	-4.189	-6.271	290.4	290.5	290.4
Amazon	1.380	1.447	1.386	13.41	13.42	13.40
Taobao	-2.010	2.489	-2.041	21.31	21.33	21.32
PUBG	-5.872	-3.040	-5.905	17.51	17.51	17.51
Reddit Politics	0.288	3.800	0.245	53.72	53.70	53.71
Reddit AskScience	2.150	5.477	2.204	189.0	188.5	188.6
Last.fm	2.856	4.539	3.069	674.4	671.3	671.5
NYC taxi	-1.295	0.428	-1.298	27.50	27.50	27.50
Stack Overflow	4.941	5.116	4.935	128.7	128.7	128.7

B.4 Density versus hazard

Outputting a density function or a hazard function over a set of intervals are similar in a very specific sense: a piecewise-constant hazard function maps deterministically to a piecewise-*exponential* density function with the same partition edges. In terms of the density function, only the within-interval shape differs. A further difference is that loss through one logit of a piecewise density function propagates to all other logits through the normalization of the softmax operation, while only outputs preceding the target time receive a loss signal in the case of the piecewise hazard function. Both approaches share the benefit of high expressive power without training instability (see Appendix E). A minor detraction from the piecewise hazard function in terms of elegance is that it must be mapped to a density function for loss calculation and for the NLL and MAE evaluations. On the other hand, the piecewise hazard is more elegant in terms of its final interval, which does not need a separate form, as a constant hazard over an infinite tail interval maps to an exponential density.

In Table 12, we compare the *pc* head to two hazard-based heads; one head uses edges from the equal-quantile based edge selection method (labeled as *qpch*) and the other head uses the likelihood-maximizing edge selection method (labeled as *pch*). The performance scores suggest that these two approaches are broadly similar in terms of effectiveness.

For both edge selection methods, switching to a piecewise hazard function yields mixed results: improved NLL in aggregate (mean rank across datasets) but consistently worse-or-equal MAE. The differences, however, are small in comparison to the differences seen when switching to another head (e.g. *logmix* or *nn*). For the equal-quantile edges, the differences are especially small. This is some evidence in support of the claim that using a piecewise hazard function leads to NLL and MAE scores broadly similar to those of a piecewise density function with the same edges.

B.5 Event sequence from a simulated Hawkes process

The datasets considered in the main text are all recorded from real-world processes. In this section we test the models on a synthetic continuous dataset—inter-event times sampled from a Hawkes

process. Table 13 shows the results for the next event prediction task on the Hawkes process dataset. Models with a *logmix* are the most performant in terms of NLL and MAE, with the *pc* head second, and the *nn* head third. These results show that the *pc* head performance does not become deleterious on truly continuous datasets.

B.5.1 DATASET GENERATION

The Hawkes process is defined by the following conditional hazard function:

$$\lambda(t|H_t) = \mu + \sum_{t_i<t} \sum_{j=1}^{M} \alpha_j \beta_j e^{-\beta_j (t-t_i)} \tag{5}$$

We follow Omi et al. (2019) and set parameters as follows: $M = 2, \mu = 0.2, \alpha_1 = 0.4, \beta_1 = 1, \alpha_2 = 0.4$, and $\beta_2 = 20$. Being synthetic, we can generate sequences of any length. We generate a sequence of $\approx 2^25$ events, but use just 2^{20} events for training, 100k for evaluation and 1M for test. The training events are taken from the beginning of the sequence, and the events for the validation and test sets and taken from the end. 10 sequences were generated (one per training repeat) each with a different random seed.

Table 13: Hawkes process dataset results. NLL and MAE results for all models. 10 training repeats were carried out, where variability is in the random dataset split, random model initialization and training stochasticity. Point estimates are means over the repeats, and variability is expressed as 95% confidence intervals calculated assuming a normal distribution of the mean: $\pm 1.96 \frac{s}{\sqrt{10}}$, where s is the sample standard deviation. The Hawkes process is defined by a conditional hazard function: $\lambda(t|H_t) = \mu + \sum_{t_i<t} \sum_{j=1}^{M} \alpha_j \beta_j \exp\{-\beta_j (t-t_i)\}$, with the parameters set as in Omi et al. (2019) to $M = 2, \mu = 0.2, \alpha_1 = 0.4, \beta_1 = 1, \alpha_2 = 0.4$, and $\beta_2 = 20$. **Bold** highlights the lowest (best) point estimate.

model	NLL ↓	MAE ↓
rnn-logmix	-0.036 ± 2.32e-03	**0.954 ± 1.56e-03**
gpt-a-logmix	**-0.036 ± 3.75e-03**	0.954 ± 2.44e-03
gpt-b-logmix	-0.035 ± 3.60e-03	0.954 ± 2.42e-03
rnn-pc	-0.029 ± 3.74e-03	0.955 ± 2.43e-03
gpt-a-pc	-0.034 ± 3.75e-03	0.954 ± 2.45e-03
gpt-b-pc	-0.031 ± 3.70e-03	0.954 ± 2.47e-03
rnn-nn	-0.015 ± 3.71e-03	0.955 ± 2.46e-03
gpt-a-nn	0.018 ± 1.01e-02	0.964 ± 2.45e-03
gpt-b-nn	-0.026 ± 4.54e-03	0.954 ± 2.78e-03
rnn-softplus	0.370 ± 3.82e-03	0.980 ± 2.50e-03
gpt-a-softplus	0.359 ± 3.87e-03	0.976 ± 2.25e-03
gpt-b-softplus	0.368 ± 4.27e-03	0.974 ± 2.96e-03
rnn-exp	0.358 ± 3.86e-03	0.976 ± 2.74e-03
gpt-a-exp	0.355 ± 4.30e-03	0.975 ± 2.13e-03
gpt-b-exp	0.358 ± 4.42e-03	0.974 ± 2.74e-03
rnn-const	0.611 ± 4.21e-03	1.084 ± 3.84e-03
gpt-a-const	0.609 ± 4.57e-03	1.085 ± 5.98e-03
gpt-b-const	0.624 ± 4.37e-03	1.070 ± 5.29e-03
thp-0	0.672 ± 2.48e-02	1.016 ± 3.24e-03
dtpp	-0.026 ± 3.85e-03	0.955 ± 2.48e-03

C DATASETS

This appendix describes the 14 datasets used throughout this work. Table 14 lists the types of events and the data sources for each dataset.

Table 14: Sources of the event sequences. The unit column reports the units used in this work, which may differ from the units used in the original source (see Appendix C.4).

Dataset	Event type	Unit	Source
Yelp airport	Time between check-ins at McCarran International Airport.	minutes	(Lüdke et al., 2023)
Yelp Mississauga	Time between check-ins at businesses in Mississauga.	minutes	(Lüdke et al., 2023)
Twitter	Time between tweets of a single user.	minutes	(Lüdke et al., 2023)
Wikipedia	Time between edits of the site's top edited pages.	minutes	(Shchur et al., 2020)
Yelp Toronto	Time between check-ins at restaurants in Toronto.	hours	(Shchur et al., 2020)
MOOC	Time between user interactions on an online course system.	minutes	(Shchur et al., 2020; Kumar et al., 2019)
Amazon	Time between user interactions on Amazon.	days [1]	(Xue et al., 2024)
Taobao	Time between user interactions on Taobao.	minutes	(Xue et al., 2024; 2022)
PUBG	Time between player deaths in games of Player Unknown's Battleground (PUBG).	seconds	(Lüdke et al., 2023)
Reddit Politics	Time between posts on the politics subreddit.	seconds	(Lüdke et al., 2023)
Reddit AskScience	Time between comments on the askscience subreddit.	seconds	(Lüdke et al., 2023)
Last.fm	Time between tracks played, per user.	minutes	(Shchur et al., 2020)
NYC taxi	Time between taxi pickup times in New York City, recorded per medallion.	minutes	(Whong, 2014)
Stack Overflow	Time between the awarding of badges on Stack Overflow, per user.	days	(Stack Overflow, 2024)

C.1 DATASET HISTOGRAMS

The histograms of event times for all of the datasets are shown in Figure 9 and Figure 10. Two x-axis ranges are used: up to the 95th percentile and up to the 60th percentile of the inter-event times. When zooming in to the 60th percentile, it is easier to see discrete components present in the distributions.

Conditional histograms in Figure 11 show the grid nature of some of the datasets. Conditional histograms in Figure 12 show how even the Stack Overflow badges dataset, which appears to have a relatively smooth distribution, also has a narrow point density associated with yearly badges.

C.2 GINI COEFFICIENT

The Gini coefficients for these datasets are listed in Table 15. The Gini coefficient is a measure of distribution inequality, which is useful in this context for indicating distributions which *might* have a mix of continuous and discrete parts. A distribution where probability is concentrated in a few isolated points in an otherwise large support will have a high Gini coefficient. The NYC Taxi dataset has such a distribution (see Figure 10g). As shown in Figure 7, the performance improvement

[1] Converted from what was estimated, by inspecting the event times, to be units of ~60 days.

Table 15: Gini coefficients for the distribution of inter-event times of the real-world datasets.

Dataset	Gini coefficient
Amazon	0.000
Stack Overflow	0.141
Yelp airport	0.148
Yelp Toronto	0.192
Yelp Mississauga	0.361
Twitter	0.438
Taobao	0.581
Last.fm	0.705
Wikipedia	0.722
PUBG	0.871
Reddit AskScience	0.877
MOOC	0.885
Reddit Politics	0.906
NYC taxi	0.926

of adding intervals to the piecewise-constant density head scales with the Gini coefficient of the dataset. This makes the Gini coefficient a useful indicator of the potential benefit of increasing the bin count of the piecewise-constant density head. The Gini coefficient is calculated without binning and operates on the resolution of the data type, float32.

C.3 Sources of the datasets

The Last.fm, MOOC, Wikipedia and Yelp Toronto datasets used in this work were those made available alongside Shchur et al. (2020). The PUBG, Twitter, Yelp Airport, Yelp Mississauga and two Reddit datasets were those made available alongside Lüdke et al. (2023). The Taobao and Amazon datasets are from Xue et al. (2024). Xue et al. (2024) also provide a small subset of NYC Taxi pickup data and a small subset of Stack Overflow badge reward data; however, we instead reverted to the original sources for these two datasets to obtain larger datasets. For the Stack Overflow dataset, reverting to the original source was also necessary to adhere to the terms of use specified by Stack Overflow (no public re-sharing). While we do not distribute the data, it can be readily obtained from the Stack Overflow website directly. The taxi pickup data was obtained from Whong (2014) and the Stack Overflow badge data was obtained from Stack Overflow (2024). From both of these sources, we draw enough sequences to reach 2^{21} inter-event times.

C.4 Inter-event time units

Many of the datasets were published with non-standard units; for example, the Taobao dataset was published in units of 10,000 seconds (~3 hours). In this work, we rescale event times of each dataset to a standard unit, such as seconds, minutes or days. The specific unit is chosen to be a unit that makes sense for each dataset (e.g. seconds or minutes are more suitable than milliseconds or days for taxi pickup intervals). Where there are multiple suitable units, we choose the unit that brings the median inter-event time closer to 10, geometrically. This latter criterion prevents the magnitude of MAE scores from differing by many orders of magnitude across datasets and makes the common magnitude easy to format. The scaling has no impact on model performance, as all models normalize their inputs and outputs.

C.5 Dataset splits

For all datasets, each experiment repeat involved randomly sampling training, validation and test sets from the full dataset, in a 6:2:2 ratio. For the Taobao and Amazon datasets, which were published with predefined splits, the splits were first merged before resampling. As most of the datasets contain heavy tails, the dataset split can have a large effect on the performance. For this reason, we believe that it is important to include the variability of performance across multiple random splits,

although we acknowledge that this prevents a direct comparison to previous works which report results targeting a single fixed split.

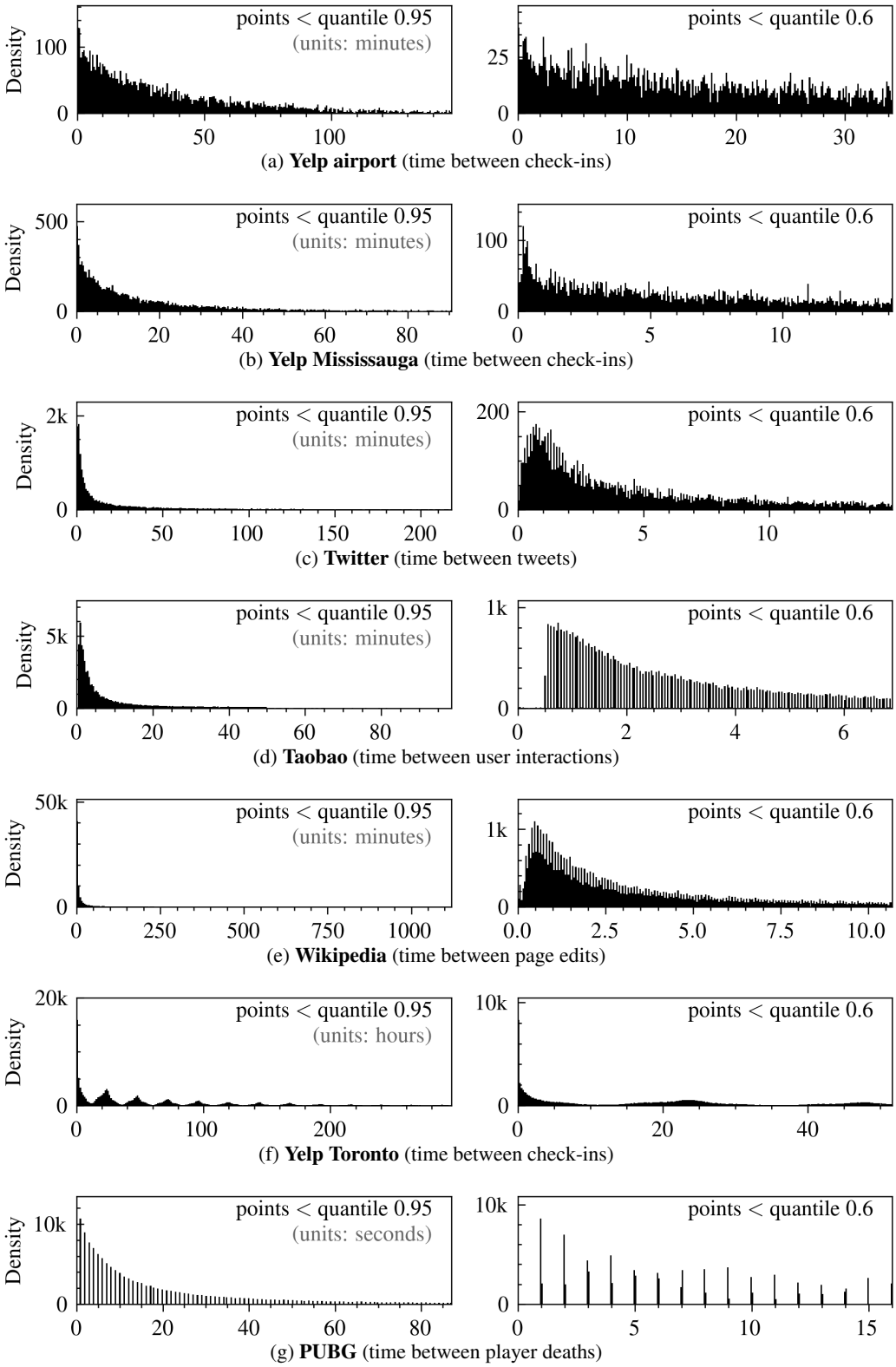


Figure 9: Histograms for 7 real-world datasets. Histograms extend until the 0.95 quantile (**left**) and 0.6 quantile (**right**). All histograms have 128 bins. The y-axis shows normalized counts.

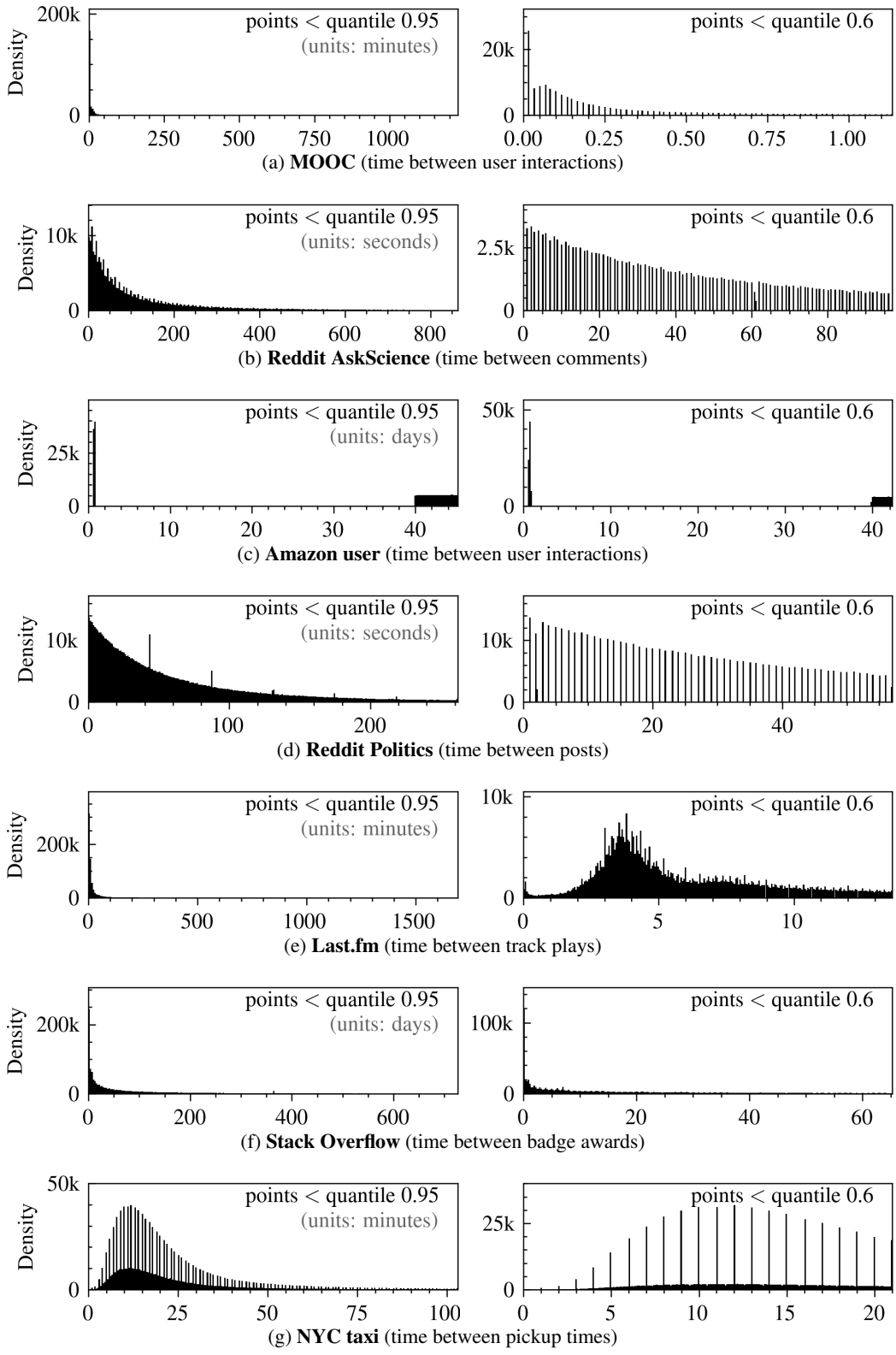
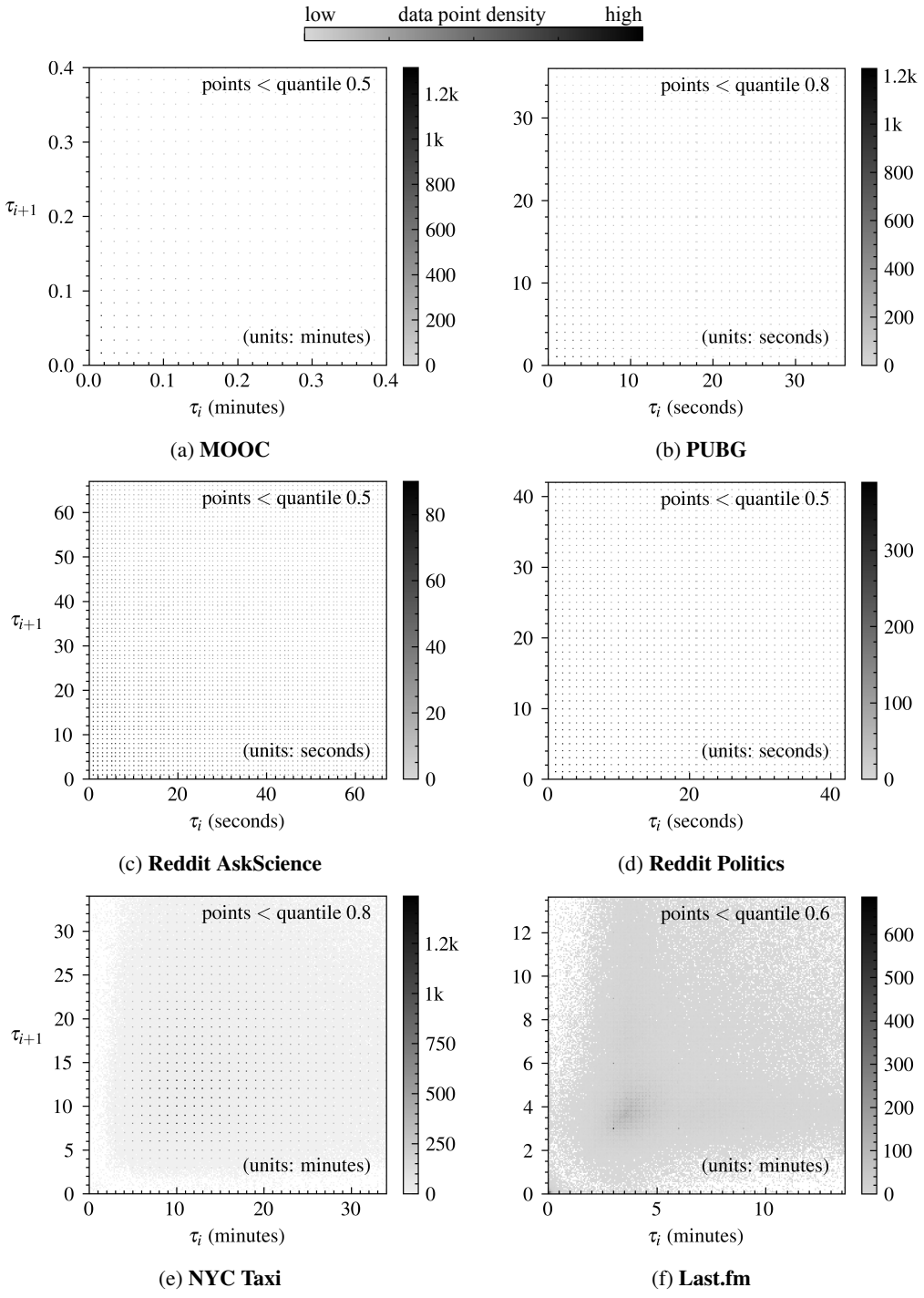


Figure 10: Histograms for 7 real-world datasets. Histograms extend until the 0.95 quantile (**left**) and 0.6 quantile (**right**). All histograms have 128 bins. The y-axis shows normalized counts.



(a) **MOOC** (b) **PUBG**

(c) **Reddit AskScience** (d) **Reddit Politics**

(e) **NYC Taxi** (f) **Last.fm**

Figure 11: Distribution of pairs of inter-event times, τ_i and τ_{i+1} for 6 real-world datasets. The first 4 datasets (**MOOC**–**Reddit Politics**) appear to have a grid structure on account of the resolution at which the data was recorded. The last 2 datasets (**NYC Taxi** and **Last.fm**) have a grid structure embedded within a smoother distribution. Each figure has a quantile threshold chosen so that the structure is visible. Each figure is a rasterization of event counts into a 256×256 pixel grid.

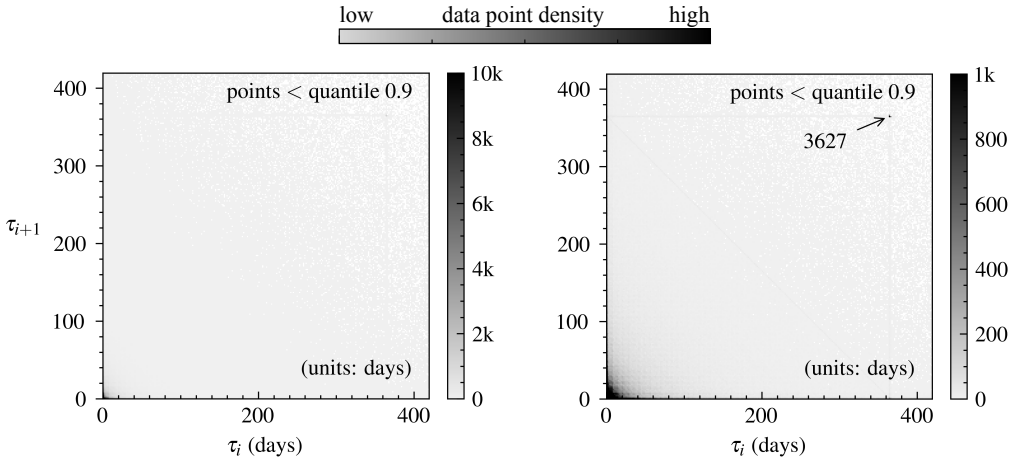


Figure 12: Distribution of pairs of inter-event times, τ_i and τ_{i+1} for the **Stack Overflow** dataset. The **right** figure is a clipped version of the **left** figure. The peak at ~365 days is small relative to the overall density, but is visible in the **right** figure when the maximum colormap value is reduced. This peak presumably corresponds to a badge typically awarded yearly.

D MODEL DETAILS

This section describes the implementation of the *rnn*, *gpt-a* and *gpt-b* stems, and the implementation of the existing heads and models used in this work.

D.1 *rnn*, *gpt-a* AND *gpt-b* STEMS

The *rnn* stem is a single-layer gated recurrent unit (GRU) (Cho et al., 2014) with 64 hidden units. The input is a 32-dimensional vector containing normalized inter-event times.

The *gpt-a* and *gpt-b* transformers follow the standard GPT-2 architecture (Radford et al., 2019) with the layer configurations (layers-heads-embedding size) set to 2-4-16 and 6-4-32 respectively. This gives *gpt-a* and *gpt-b* 108k and 1.20M trainable parameters, respectively. The standard GPT-2 small configuration defined by Radford et al. (2019) uses a 12-12-768 configuration, highlighting that both *gpt-a* and *gpt-b* are very small in the context of GPTs used as large language models. Computational limits prevented the investigation of larger models.

D.2 INPUT ENCODING FOR *gpt* BASED MODELS

For the *gpt-a* and *gpt-b* stem architectures, the scalar values representing inter-event times are converted to a vector representation according to Algorithm 2 before being input into the transformers. This approach is similar to that used by Zuo et al. (2020). A point of difference is that we do not hard-code the sinusoidal frequencies but choose appropriate scales for each dataset so that both long and short intervals of each dataset are captured.

Algorithm 2 Encode a scalar value v into a vector of even size n. v_{epsilon} should be chosen near the smallest inter-event time expected to be encountered and v_{max} should be chosen to be near the largest time expected to be spanned by inputs to the model, and will depend on the length of a model's input. In this work, we choose v_{epsilon} as the smallest inter-event time observed in the training set, and v_{max} as the largest time between events t_i and t_{i+L} where L is the input length of the model.

Function encodeValue$(v, v_{\text{epsilon}}, v_{\text{max}}, n)$
  $\text{scale}_{\text{min}} \leftarrow 2\pi / v_{\text{max}}$ {Or use $\frac{3}{2}\pi$ to add a little more range}
  $\text{scale}_{\text{max}} \leftarrow 2\pi / v_{\text{epsilon}}$
  $\text{scale} \leftarrow \text{linspace}(\log(\text{scale}_{\text{min}}), \log(\text{scale}_{\text{max}}), n/2)$ {A vector of length $n/2$}
  $x_1 \leftarrow \cos(\exp(\text{scale}) \cdot v)$
  $x_2 \leftarrow \sin(\exp(\text{scale}) \cdot v)$
  $x \leftarrow \text{concat}(x_1, x_2)$
  return x

D.3 IMPLEMENTATION OF EXISTING MODELS

The *const*, *exp* and *nn* heads are implemented following the code accompanying (Omi et al., 2019). The *logmix* head is implemented following the code accompanying Shchur et al. (2020). The *thp* model is implemented partially following (Zuo et al., 2020), but deferring to the implementation accompanying (Xue et al., 2024) for a number of bug fixes to the original model. The *dtpp* model is implemented following the code accompanying Panos (2024).

One deviation from these implementations is in model initialization. In order to improve convergence speed and stability while training, all models were initialized so that their outputs have first and second moments similar to those of the training set's empirical distribution.

E LIMITATIONS OF EXISTING OUTPUT REPRESENTATIONS

The existing output structures considered in this work have limitations in terms of distribution flexibility and training stability, both of which contribute to preventing them from representing multimodal distributions with very narrow peaks.

The constant, exponential and softplus intensity heads have limited distribution flexibility. The *logmix* head has much greater capacity to represent complex distributions; however, it experiences training instability in the presence of narrow peaks in the target distribution. The *nn* head can also represent complex distributions; however, it appears to have limited ability to output different complex distributions for different inputs.

E.1 DISTRIBUTION FLEXIBILITY

The constant intensity, exponential intensity and the softplus intensity correspond to distributions over $[0, \infty)$ with 1 or 2 parameters. With only 1 or 2 parameters, they do not have the flexibility to represent distributions where there are multiple separated peaks of high probability density, which is a property of several of the real-world datasets (see Figure 11).

The constant intensity output corresponds to the exponential distribution. The exponential distribution has a single scalar parameter with the distribution's mode fixed at 0. The distribution cannot concentrate probability at any non-zero point.

The exponential intensity output corresponds to the Gompertz distribution. The Gompertz distribution has two parameters, a scale and a shape parameter. It is a unimodal distribution so cannot concentrate probability at two or more separated points.

The softplus intensity function corresponds to a unimodal probability distribution with two shape parameters. Its unimodal shape is inferred from the monotonicity of the softplus function: a monotonic intensity implies a convex cumulative intensity, and both then imply a unimodal density function. Being unimodal, it is restricted in its ability to concentrate probability at multiple separated points. The distribution lacks a familiar name and does not have an elementary closed form (it can be expressed in terms of a polylogarithm). Rather than being motivated by a particular distributional family, the softplus function is used as it is strictly positive and smooth (Mei & Eisner, 2017; Zuo et al., 2020).

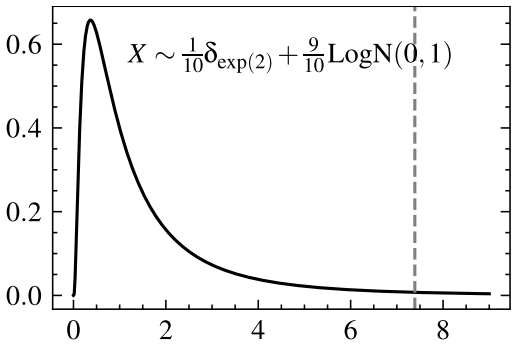
For these three heads, a theoretical argument is sufficient to explain how they cannot represent distributions with many peaks. For the *nn* and *logmix* heads, which have much greater capacity to represent complex distributions, we present hypotheses for limitations along with supporting empirical evidence.

E.2 MIXTURE OF LOGNORMALS INSTABILITY

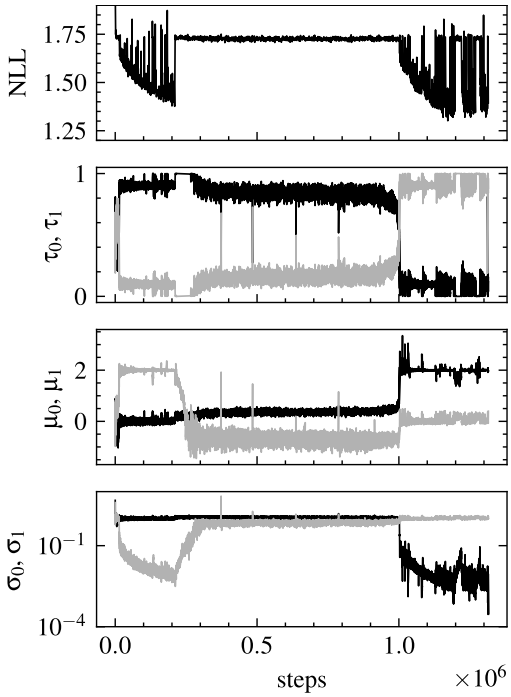
A mixture of lognormals can theoretically represent distributions with multiple sharp peaks. However, when training models with this output structure on datasets where there is a significant discrete component to the inter-event distributions, we do not observe the models concentrating probability in narrow peaks to the same extent as can be seen with the piecewise-constant model; see the densities in Figure 4, for example. We argue that such models struggle to represent such distributions on account of training instability in the presence of singularities. A mixture of lognormals on $t \in (0, \infty)$ is equivalent to a mixture of Gaussians on $y \in (-\infty, \infty)$, where $y = \log(t)$, and so models that output a mixture of lognormals can be thought of as outputting a mixture of Gaussians on a log scale. A mixture of Gaussians can produce any number of modes, up to the number of components in the mixture.

The process of fitting these mixtures using the expectation-maximization (EM) algorithm demonstrates how this flexibility does not necessarily allow the mixture to represent distributions with sharp peaks. When fitting a mixture of Gaussians using the EM algorithm, a component's variance can collapse toward zero at a single point. To avoid this, strategies such as introducing priors on the parameters or resetting the EM optimization procedure are used (Bishop, 2006).

Although our models are trained with stochastic gradient descent rather than EM, the singularity in the log-likelihood function as $\sigma \to 0$ is a property of the objective function, not the optimizer,



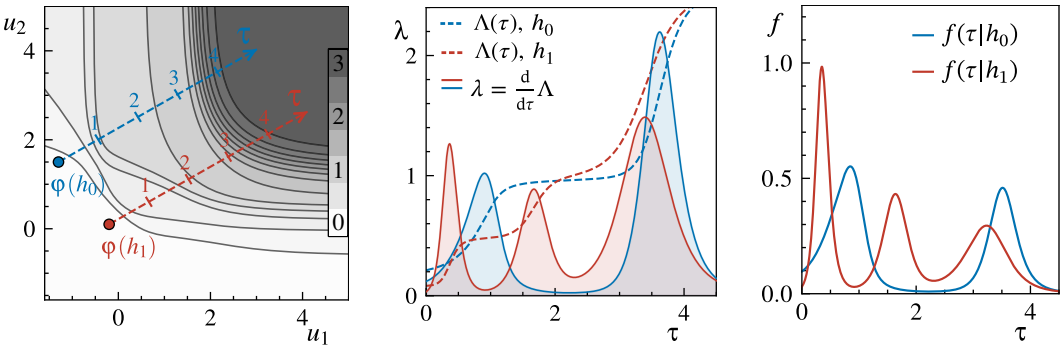
(a) The probability distribution of a dataset generated from a mixture of a point mass at $t = \exp(2) \sim 7.4$ and a lognormal distribution with parameters $\mu = 0$ and $\sigma = 1.0$.



(b) How the NLL (evaluated on a held-out validation set) and the model's six parameters change while training.

Figure 13: The instability of a logmix output head when a dataset has a probability mass at a certain point. **(a):** shows the distribution of inter-event times. The dataset is formed from a mixture where 10% of values come from a point source at $t = \exp(2)$, and the remaining values are drawn from a lognormal distribution with parameters $\mu = 0$ and $\sigma = 1.0$. All events are independent. The model used is a *gpt-a* stem with a 2-component lognormal mixture output. The model is trained using stochastic gradient descent with momentum (0.9) and a learning rate of 5×10^{-4}. **(b):** shows the mixture parameters (the model output) and the NLL (calculated on an evaluation set) over the course of training. The six model parameters are the mixing parameters (τ_0, τ_1) and the means and variances ($\mu_0, \mu_1, \sigma_0, \sigma_1$) of the two underlying Gaussian distributions. Notably, neither μ_0 nor μ_1 approach 2.0 in a stable manner. Training stopped when evaluating the distribution median produced `NaN` values. This instability occurs reliably across runs.

and we hypothesize that stochastic gradient descent encounters it. If this is the case, we would expect to see training instability in the form of sudden loss spikes and parameter perturbations near when a component's variance parameter approaches zero. A simple case study (Figure 13) shows *gpt-a*, outputting a 2-component lognormal mixture, trained on a dataset generated from a mixture of a lognormal distribution and a point mass. As one of the output components narrows around the single point, sudden spikes in gradient cause the parameters of the component to be perturbed and its mixing weight to suddenly decrease. "Real-world" datasets such as the NYC taxi dataset have such point masses, and training stability in the presence of these sharp distributions is a candidate explanation for why the *logmix* head is not observed to concentrate probability at such points in a stable manner. To prevent the instability, regularization could be introduced in the form of a prior on the variance of each component. Doing so amounts to specifying a resolution of interest for the task and would be evidence of the benefit of acknowledging the discrete nature of recorded event times.



(a) The cumulative intensity, Λ, evaluated for a square of u-space. Values are rounded to the nearest integer for visualization purposes.

(b) Λ and its derivative along the two τ-paths from **(a)**, which are $W_\tau \tau + \phi(h)$ for h_1 and h_2.

(c) The probability densities corresponding to the two paths in **(a)**.

Figure 14: An `nn` head where the first embedding vector $\mathbf{u}$ is 2-dimensional. The matrix projecting τ into $\mathbf{u}$-space, $W_\tau \in \mathbb{R}^{2\times 1}$, is the unit vector $[0.86, 0.51]^T$. The second and third layers have 5 and 3 units respectively ($W_1 \in \mathbb{R}^{5\times 2}, W_2 \in \mathbb{R}^{5\times 3}, W_{\text{out}} \in \mathbb{R}^{3\times 1}$). **(b)** shows the altitude change along the τ-paths from **(a)**; this is the cumulative intensity Λ. The derivative, λ, is a solid curve. The corresponding probability densities are shown in **(c)**. All possible probability densities for this head correspond to a different starting position in the u-space.

E.3 THE *nn* HEAD CAN EASILY TRANSLATE BUT NOT MODULATE PEAKS

The *nn* head from Omi et al. (2019) is, in principle, a flexible output structure. However, the results in the main body suggest there is some limitation that prevents it from achieving NLL scores similar to those of the *logmix* or *pc* heads. We find, empirically, that the *nn* head can successfully model sharp peaks for a stationary distribution, but struggles to switch these peaks on and off for different inputs. Inspecting the implementation of the *nn* head suggests a mechanism for this phenomenon. Next, we outline the implementation of the *nn* head. Then we start by presenting the theoretical argument along with empirical evidence.

E.3.1 *nn* HEAD IMPLEMENTATION

The *nn* head has two inputs: τ, the inter-event time being evaluated, and $\mathbf{h}$, the hidden state vector emitted by the RNN stem. The output of the *nn* head is a scalar value representing the cumulative hazard function, $\Lambda(\tau)$ evaluated at the input τ and conditional on the history representation, $\mathbf{h}$. The hazard function, $\lambda(\tau)$, is calculated from automatic differentiation of $\Lambda(\tau)$ with respect to the input τ. As Λ must be a non-decreasing function of τ in order to be a valid cumulative hazard function, the multiplicative weights in the τ pathway of the *nn* head are constrained to be non-negative.

The relationship between input and output can be described as follows. Let $W_\tau \in \mathbb{R}^{d\times 1}$, $W_h \in \mathbb{R}^{d\times d_{\text{RNN}}}$ and $\mathbf{b_h} \in \mathbb{R}^d$ be the first weights in the network that map the input pair $(\tau, \mathbf{h})$ to the first embedding vector $\mathbf{u} \in \mathbb{R}^d$. It will be useful, as shorthand, to define the function $\phi(\mathbf{h}) = W_h\mathbf{h} + \mathbf{b_h}$. The remaining layers, of which there are 3, are linear layers with two tanh activations and a final softplus activation. The input-output relationship can be written as:

$$
\begin{aligned}
\phi(\mathbf{h}) &= W_h\mathbf{h} + \mathbf{b_h} \\
\mathbf{u} &\leftarrow \tanh(W_\tau\tau + \phi(\mathbf{h})) \\
u_{\text{final}} &\leftarrow W_{\text{out}} \tanh(W_2 \tanh(W_1\mathbf{u} + \mathbf{b_1}) + \mathbf{b_2}) + b_{\text{out}} \\
\Lambda &\leftarrow \text{softplus}(u_{\text{final}})
\end{aligned}
\tag{6}
$$

Where W_τ, W_1, W_2, and W_{out} matrices have non-negative elements.

E.3.2 PARALLEL PATHS IN A FIXED, CLIMBING, TERRACED LANDSCAPE

The *nn* head can be described as being only able to map time points to a line through a fixed landscape. A consequence of the *nn* head's structure is that for a given history representation, $\mathbf{h}$, the inter-event time, τ, defines a straight path in the $\mathbf{u}$-space whose image is non-decreasing. Figure 14 illustrates a toy *nn* head where the $\mathbf{u}$-space is 2-dimensional; the following layers have 5 and 3 units respectively. The first sub-figure, Figure 14a, is a contour plot showing the image through the network of a square of $\mathbf{u}$-space. The parameters that map the pair $(\tau, \mathbf{h})$ to $\mathbf{u}$-space are fixed once learned (W_τ, W_h and $\mathbf{b}_h$), and the only way to control the path taken through $\mathbf{u}$-space is through the vector $\mathbf{h}$ emitted from the model stem. As W_τ is fixed, all possible paths are parallel lines. The middle figure, Figure 14b, plots the altitudes of the landscape in Figure 14a, which is the cumulative hazard $\Lambda(\tau)$, along two example paths, starting at $\phi(\mathbf{h_1})$ and $\phi(\mathbf{h_2})$. The resulting probability density functions are shown in Figure 14c. The real *nn* head has a 64-dimensional $\mathbf{u}$-space, rather than a 2-dimensional one; however, the same limitation that, once trained, all possible probability distributions correspond to parallel paths in $\mathbf{u}$-space, still applies. It is clear how moving $\phi(\mathbf{h})$ along the W_τ direction can translate the output distribution. Moving $\phi(\mathbf{h})$ in a different direction slices through a different region of the landscape and can change the distribution's shape; however, the change is constrained by whatever shape variations the fixed landscape contains, and so many distribution shapes are not reachable from any $\phi(\mathbf{h})$.

We presented the toy model as a way to give an intuitive picture of the *nn* head's possible limitations—the toy model does not act as conclusive evidence for any specific limitation. The next section presents empirical evidence which (partially) supports the hypothesis that the *nn* head can easily translate but not modulate peaks in a distribution.

E.3.3 EMPIRICAL EVIDENCE

We take a closer look at the behaviour of the *nn* head by looking at the output of the *rnn-nn* model trained on four datasets (1 real, 3 synthetic). We also include the outputs of *rnn-pc* and *rnn-logmix* for comparison.

Narrow densities in a real-world dataset. First, we look at how well the *rnn-nn* model's output lines up with the empirical distribution for the PUBG dataset, a dataset with multiple narrow peaks in the distribution of inter-event times (see Figure 9g). 10 models (1 per run) were trained for the results of the main body (Section 6), and we take 1 of these and inspect its output. We present just 1 model here, but we have checked all 10 and they show the same pattern that we present below. In Figure 15, the cumulative distribution functions (CDFs) corresponding to the output emitted by the *rnn-nn*, *rnn-pc* and *rnn-logmix* models trained on the PUBG dataset are compared to the training set's empirical CDF. For visual comparison, the CDF rather than the PDF is used as the PDFs of the models have a disparate range of values. The model CDFs are calculated by averaging over 4096 input histories from the training set. What is apparent from Figure 15 is that jumps in the CDF corresponding to narrow peaks in the PUBG distribution are matched, albeit roughly, by the *rnn-logmix* and *rnn-pc* models, but not by the *rnn-nn* model whose CDF remains smooth. This is an example where the *rnn-nn* model fails to reflect narrow peaks present in the target distribution. The next case study shows that the *rnn-nn* model *does* have the capacity to represent narrow peaks, if the target distribution is stationary.

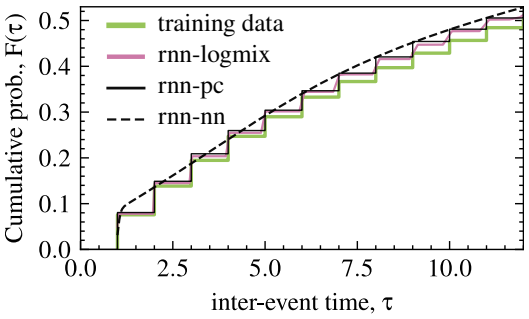
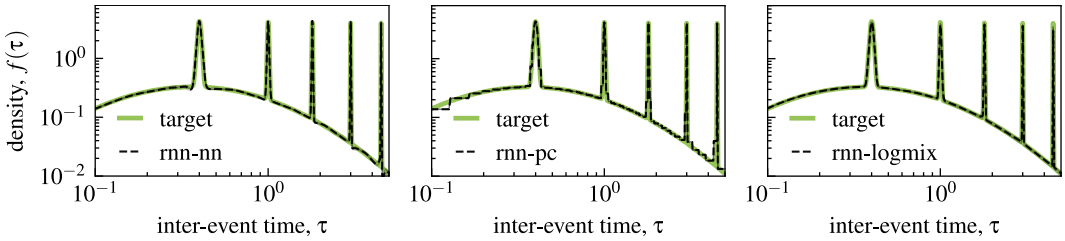


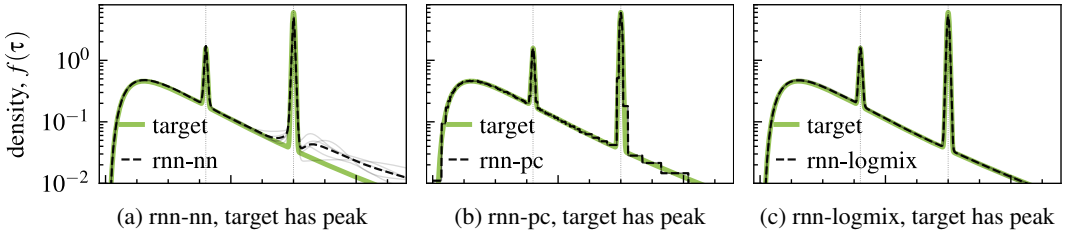
Figure 15: Empirical and predicted cumulative distribution function (CDF) for the PUBG dataset. The green curve is the empirical CDF of the training set's inter-event times. The other curves are mean CDFs calculated from `rnn-nn`, `rnn-logmix` and `rnn-pc` outputs, meaned over 4096 input histories drawn from the training set.

Fixed peaks. We use a synthetic dataset to show a case where the *rnn-nn* model can be trained to fit a distribution with multiple sharp peaks. Results are shown in Figure 16. A synthetic sequence's inter-event times are independently drawn from a six-component mixture: a lognormal distribution



(a) rnn-nn (b) rnn-pc (c) rnn-logmix

Figure 16: Probability distributions emitted by *rnn-nn*, *rnn-pc* and *rnn-logmix* models trained on a synthetic sequence where inter-event times are drawn from a stationary mixture (lognormal and five narrow Gaussian peaks). The true distribution of the inter-event times is shown in green. The *rnn-nn*'s distribution contains five peaks at the correct locations. A black dashed line is the mean of five training runs. The individual runs are shown as semi-transparent gray curves; however, all are close to the mean and difficult to distinguish.



(a) rnn-nn, target has peak (b) rnn-pc, target has peak (c) rnn-logmix, target has peak

(d) rnn-nn, target without peak (e) rnn-pc, target without peak (f) rnn-logmix, target without peak

Figure 17: Two different outputs for three models trained on a synthetic sequence where the distribution of the next inter-event time switches between having and not having a narrow peak. The distribution of τ_{i+1} depends only on τ_i: if $\tau_i \geq 1$, then $p(\tau_{i+1}|\tau_i) \sim \text{lognormal}(0, 1)$, else $p(\tau_{i+1}|\tau_i)$ is a mixture of the same lognormal distribution and a narrow Gaussian ($\sigma = 0.02$) centered at $\tau_{i+1} = 2$. The mixture is $0.7 \cdot \text{LogNormal}(0, 1) + 0.3 \cdot \mathcal{N}(2, 0.02^2)$. Both *rnn-pc* and *rnn-logmix* can (somewhat) use the history input to modulate the probability density at the peak location. We do not observe the *rnn-nn* model carrying out this modulation. This experiment was repeated five times with different random seeds, and the same pattern was observed each time (the figures above show just one run).

and five narrow Gaussian distributions. The inter-event times do not depend on the event history. This case study is evidence that the *nn* head can represent distributions with multiple narrow peaks. Interestingly, the *rnn-logmix* model struggles to fit all five peaks.

Present/absent peaks. Modulating peaks is not as well achieved. Now we use a synthetic sequence with a smooth inter-event time distribution with a single peak; but this time, the peak is either present or absent depending on the input. If the most recent inter-event time, τ_i is less than 1, then the peak is present in the distribution, otherwise it is absent. Figure 17 shows the results of training *rnn-nn* on this dataset, along with *rnn-pc* and *rnn-logmix*. Both *rnn-pc* and *rnn-logmix* effectively modulate the density at the location of the peak, whereas *rnn-nn*'s predictions underestimate when the peak is

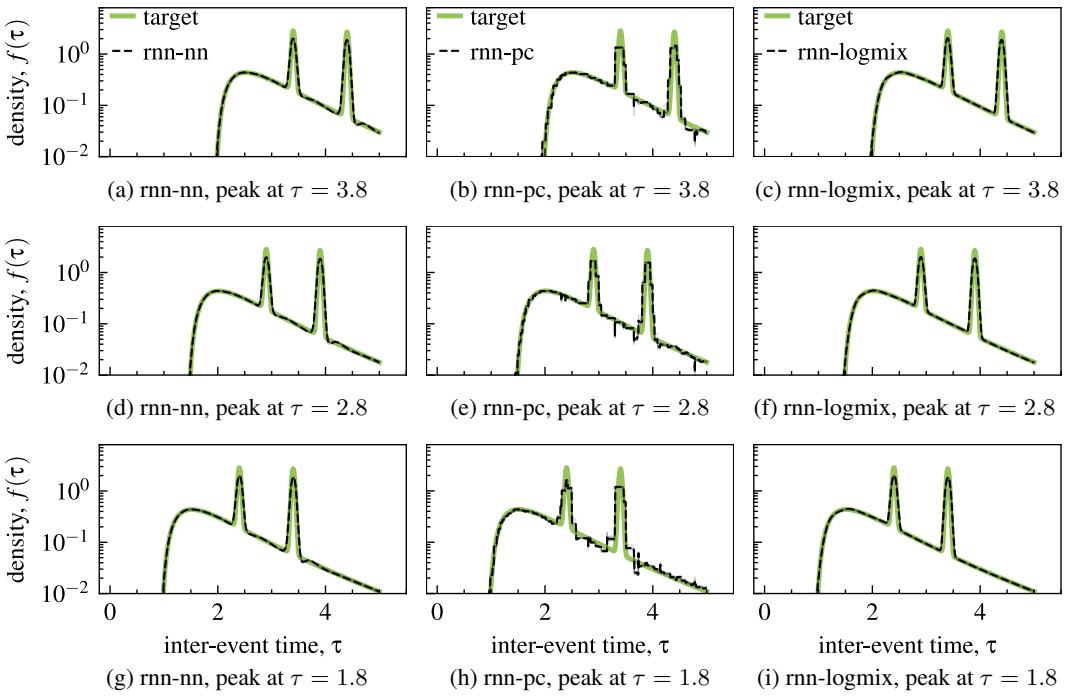
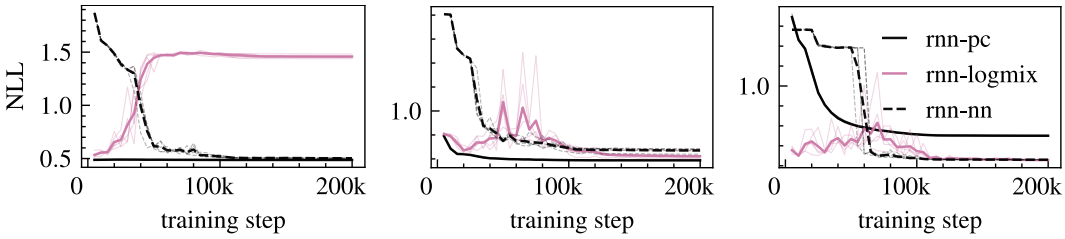


Figure 18: Three models trained on a synthetic sequence in which the distribution of the next inter-event time has a narrow peak whose center is equal to the previous inter-event duration. The distribution of τ_{i+1} is a mixture of a background lognormal and a Gaussian ($\sigma = 0.03\tau_i$) centered at τ_i. Specifically, $p(\tau_{i+1} \mid \tau_i) = 0.6 \cdot \text{LogNormal}(0, 1) + 0.4 \cdot \mathcal{N}\big(\tau_i,\, (0.03\,\tau_i)^2\big)$. All three models can use the history input to translate the location of a peak.

present and overestimate when the peak is absent. Peak/non-peak density is $\sim 25\times$ for the *rnn-pc* and *rnn-logmix* models and only $\sim 2\times$ for the *rnn-nn* model. This case study is evidence (though not conclusive) that the *nn* head struggles to modulate sharp peaks in a distribution, even though it can represent both in the case of a stationary distribution. It is tempting to connect this failure to the idea put forward by the toy model of Figure 14: that both distributional shapes may not be present in the $\mathbf{u}$-space along the fixed W_τ direction, or that they are present but the model struggles to learn to move $\phi(\mathbf{h})$ between them. We do not have enough evidence to support this connection and instead simply state it as a hypothesis.

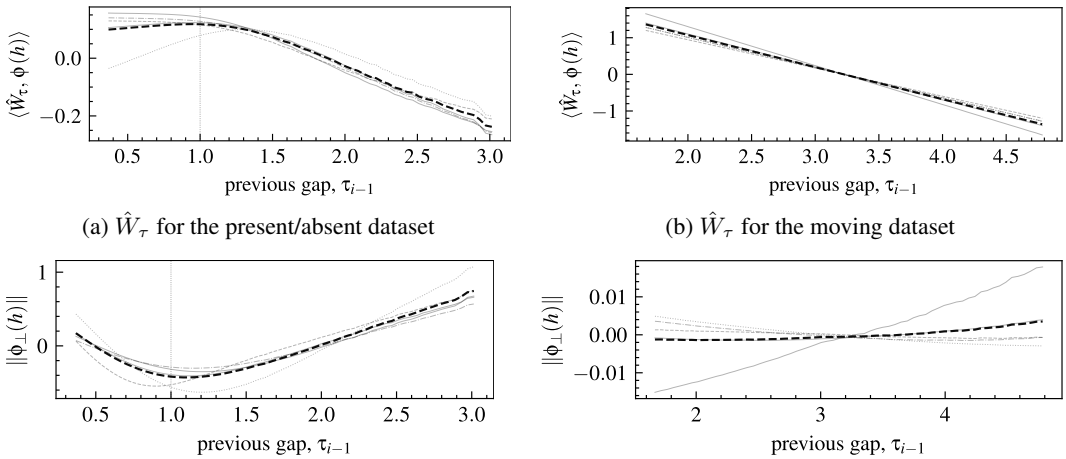
Translated peaks. A third case study, similar to the second, considers the *translation* of a single peak in an otherwise smooth distribution, where the amount of translation depends on the input. The output distribution of the next inter-event time τ_{i+1} for this synthetic event sequence is a lognormal and Gaussian mixture, where the narrow Gaussian is centered at a copy of the previous inter-event time, τ_i. The training set contains a range of τ_i values between 0 and 10. Figure 18 shows three example outputs for *rnn-nn*, *rnn-pc* and *rnn-logmix*. For this dataset, the peak is not narrow enough to cause training instability for the *logmix* head. In this case, *rnn-nn* and *rnn-logmix* perform similarly, with the output containing a peak that translates with input. It is the *rnn-pc* model that struggles the most in this case. This case study is evidence that the *nn* head *can* translate peaks in the output distribution conditioned on input history. If we were to explain this result using the toy model of Figure 14, we could argue that a translated distribution must be present in the $\mathbf{u}$-space, and that a movement of $\phi(\mathbf{h})$ in the W_τ direction would enable this translation. We do not have enough evidence to support this connection, and present it only as a candidate explanation.

Training details. For the three case studies above, the three models, *rnn-nn*, *rnn-pc* and *rnn-logmix*, were trained for 200k steps with a batch size of 64, with all other training settings the same as the main body (see Appendix F). The checkpoints with the best loss were used for evaluation. Each



(a) fixed (b) present/absent (c) translated

Figure 19: NLL evaluated every 5k steps throughout 200k-step training runs for the three models (*rnn-nn*, *rnn-pc* and *rnn-logmix*) on the three synthetic datasets (fixed peaks, binary peaks, and moving peaks). The NLL is calculated on 2^{16} samples from a held-out set. There are 5 individual runs displayed as semi-transparent curves, and the mean across them is opaque.



(a) $\hat{W}_\tau$ for the present/absent dataset (b) $\hat{W}_\tau$ for the moving dataset

(c) $W_\tau^\perp$ for the present/absent dataset (d) $W_\tau^\perp$ for the moving dataset

Figure 20: Decomposition of $\phi(h) \in \mathbb{R}^{64}$ (from Equation (6)) along the unit direction $\hat{W}_\tau = W_\tau / \|W_\tau\|$, and the (63-dimensional) orthogonal complement, $W_\tau^\perp$. This is done for two datasets, the *present/absent peaks* dataset (**left**) and the *translated peaks* dataset (**right**). The component of $\phi(h)$ parallel to $\hat{W}_\tau$ is $\langle \hat{W}_\tau, \phi(h) \rangle$ (**top**) and the norm of the 63-d orthogonal component is $\|\phi(h) - \langle \hat{W}_\tau, \phi(h) \rangle \hat{W}_\tau\|$, which we denote as $\|\phi_\perp(h)\|$ (**bottom**). For each dataset we draw 2^{17} histories from a held-out set and evaluate $\phi(h)$, making note of τ_{i-1}. This is done for 5 runs (grey), and the mean across runs is plotted in black. All runs have their individual mean subtracted to bring the curves into the same range. The per-run mean operates on values binned by the x-axis, τ_{i-1} into 64 bins from the 5th until the 95th quantile of τ_{i-1}. The parallel component is strongly correlated with τ_{i-1} for the *translated peaks* dataset, and not for the *present/absent peaks* dataset.

experiment was repeated five times to ensure the results were consistent. We present just one run for each experiment to keep the figures clear. Training curves for all runs are shown in Figure 19.

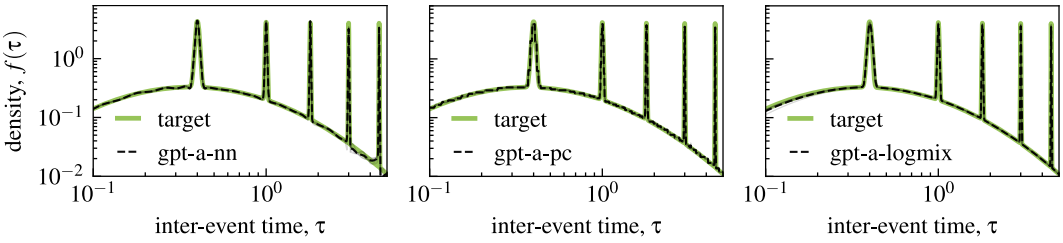
E.3.4 Discussion

The test cases and the toy model suggest (but do not conclusively prove) that the *nn* head can easily translate peaks in a distribution, but is less capable of making point changes to a distribution such as adding and removing a narrow peak. We suggest that this may explain why the *nn* head is outperformed by the *logmix* and *pc* heads on many of the datasets considered in this work. We have presented only limited evidence to support this implication, and so we present it as a theory rather than a conclusion.

This section focused on the *nn* head; however, some of the results are relevant to the *logmix* head as well. The *rnn-logmix* training loss curve in Figure 19a shows how the model's loss increases and doesn't recover. Inspecting the output distribution of a model checkpoint after this spike shows that the model loses the ability to represent all of the peaks in the distribution. This result supports the previous section's argument that the *logmix* head can be perturbed while training if there are narrow peaks in the target distribution. The training curves for the two other datasets (present/absent and translated peaks) also suggest instability in the training process.

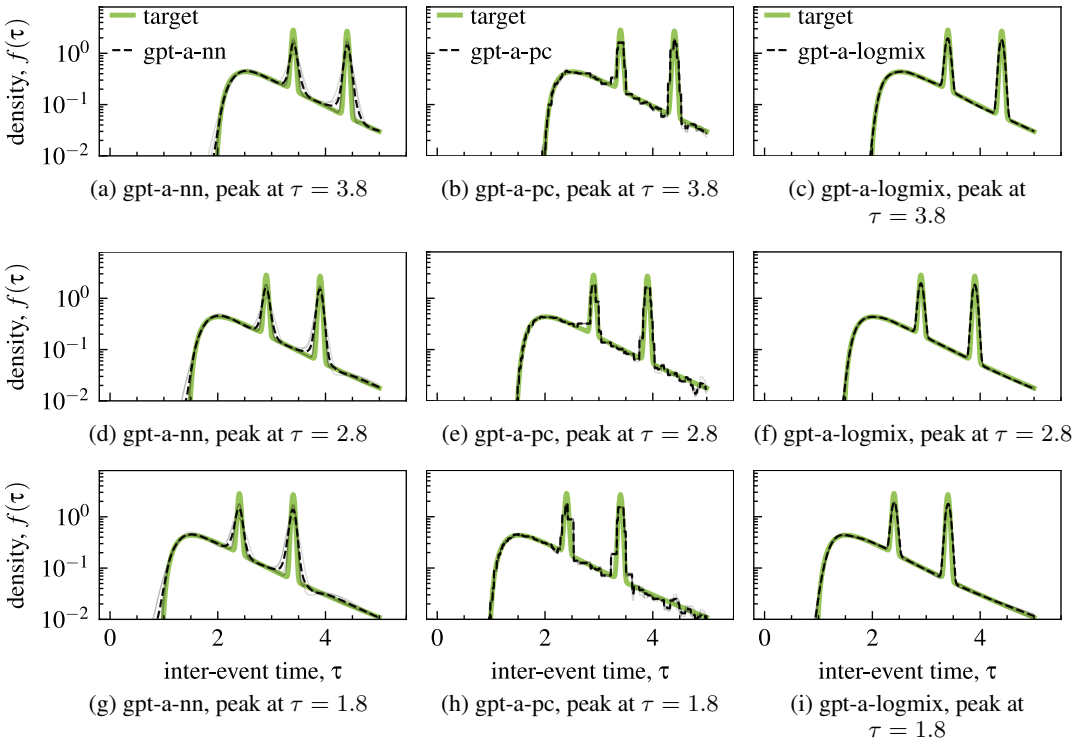
E.3.5 Model ablation: repeat for *gpt-a* stem

The synthetic experiments in the above section used the RNN stem for all models. To rule out the possibility that the results are peculiar to the RNN stem, the same experiments were repeated with the *gpt-a* stem, and those results appear below.



(a) gpt-a-nn (b) gpt-a-pc (c) gpt-a-logmix

Figure 21: Probability distributions emitted by *gpt-a-nn*, *gpt-a-pc* and *gpt-a-logmix* models trained on a synthetic sequence where inter-event times are drawn from a stationary mixture (lognormal and five narrow Gaussian peaks). The true distribution of the inter-event times is shown in green. The *gpt-a-nn*'s distribution contains five peaks at the correct locations. A black dashed line is the mean of five training runs. The individual runs are shown as semi-transparent gray curves; however, all are close to the mean and difficult to distinguish.

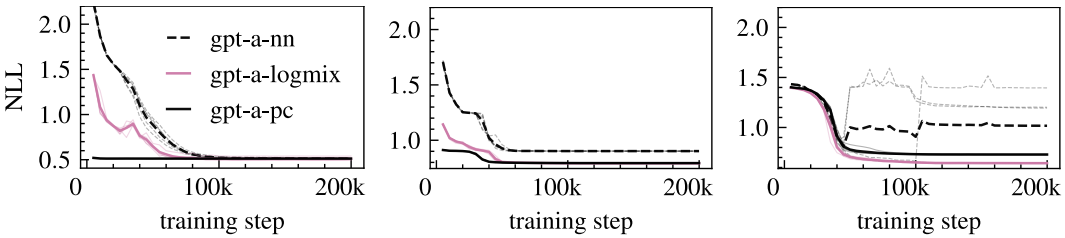


(a) gpt-a-nn, peak at $\tau = 3.8$ (b) gpt-a-pc, peak at $\tau = 3.8$ (c) gpt-a-logmix, peak at $\tau = 3.8$

(d) gpt-a-nn, peak at $\tau = 2.8$ (e) gpt-a-pc, peak at $\tau = 2.8$ (f) gpt-a-logmix, peak at $\tau = 2.8$

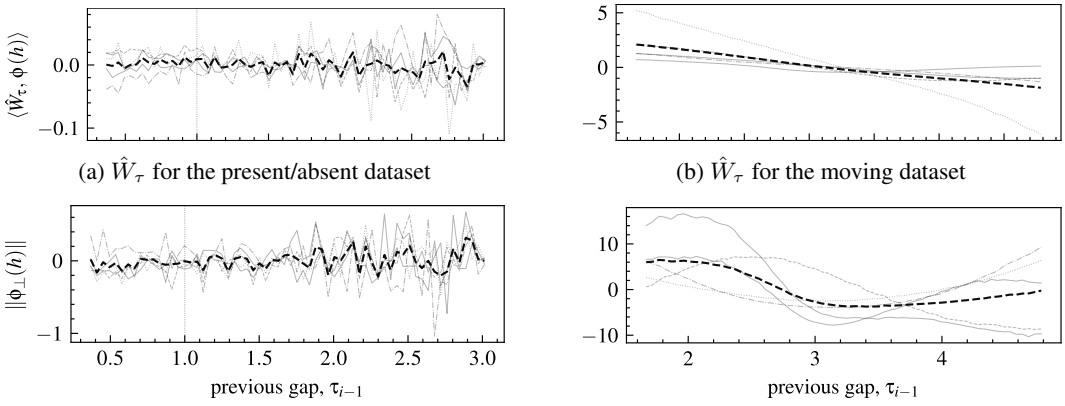
(g) gpt-a-nn, peak at $\tau = 1.8$ (h) gpt-a-pc, peak at $\tau = 1.8$ (i) gpt-a-logmix, peak at $\tau = 1.8$

Figure 22: Three models trained on a synthetic sequence in which the distribution of the next inter-event time has a narrow peak whose center is equal to the previous inter-event duration. The distribution of τ_{i+1} is a mixture of a background lognormal and a Gaussian ($\sigma = 0.03\tau_i$) centered at τ_i. Specifically, $p(\tau_{i+1} \mid \tau_i) = 0.6 \cdot \mathrm{LogNormal}(0, 1) + 0.4 \cdot \mathcal{N}\big(\tau_i,\ (0.03\,\tau_i)^2\big)$. All three models can use the history input to translate the location of a peak.



(a) fixed (b) present/absent (c) translated

Figure 23: NLL evaluated every 5k steps throughout 200k-step training runs for the three models (*gpt-a-nn*, *gpt-a-pc* and *gpt-a-logmix*) on the three synthetic datasets (fixed peaks, binary peaks, and moving peaks). The NLL is calculated on 2^{16} samples from a held-out set. There are 5 individual runs displayed as semi-transparent curves, and the mean across them is opaque.



(a) $\hat{W}_\tau$ for the present/absent dataset

(b) $\hat{W}_\tau$ for the moving dataset

(c) $W_\tau^\perp$ for the present/absent dataset

(d) $W_\tau^\perp$ for the moving dataset

Figure 24: Decomposition of $\phi(h) \in \mathbb{R}^{64}$ (from Equation (6)) along the unit direction $\hat{W}_\tau = W_\tau / \|W_\tau\|$, and the (63-dimensional) orthogonal complement, $W_\tau^\perp$. This is done for two datasets, the *present/absent peaks* dataset (**left**) and the *translated peaks* dataset (**right**). The component of $\phi(h)$ parallel to $\hat{W}_\tau$ is $\langle \hat{W}_\tau, \phi(h) \rangle$ (**top**) and the norm of the 63-d orthogonal component is $\|\phi(h) - \langle \hat{W}_\tau, \phi(h) \rangle \hat{W}_\tau\|$, which we denote as $\|\phi_\perp(h)\|$ (**bottom**). For each dataset we draw 2^{17} histories from a held-out set and evaluate $\phi(h)$, making note of τ_{i-1}. This is done for 5 runs (grey), and the mean across runs is plotted in black. All runs have their individual mean subtracted to bring the curves into the same range. The per-run mean operates on values binned by the x-axis, τ_{i-1} into 64 bins from the 5th until the 95th quantile of τ_{i-1}. The parallel component is strongly correlated with τ_{i-1} for the *translated peaks* dataset, and not for the *present/absent peaks* dataset.

F TRAINING AND EVALUATION DETAILS

This appendix describes hyperparameters and other settings associated with training and evaluation. It also describes the compute resources used.

Software. PyTorch 2.6.0 with CUDA 13.2 was used for training and evaluation.

Optimizer. All training runs use the AdamW optimizer described by Loshchilov & Hutter (2019). AdamW parameters (β_1, β_2, eps, weight decay) were set to (0.9, 0.99, 1×10^{-5}, 0.02)—chosen roughly in line with heuristics described by Howard & Gugger (2020). Weight decay was applied to all parameters except bias and embedding parameters.

Precision. PyTorch's automatic mixed precision and gradient scaling were used during training. In cases where greater numerical precision is needed, such as when calculating probabilities from hazard outputs, 32-bit floating-point precision was used.

Loss function. All models, except one, were trained to minimize negative log-likelihood. The exception is *thp* which uses a loss term that is the sum of a negative log-likelihood term and a mean-squared error term (Zuo et al., 2020).

Model choice. The final models are selected from the checkpoints taken during training with the lowest validation loss.

Learning rate scheduler. The 1-cycle learning rate policy with 3 phases described by Smith & Topin (2019) was used.

Maximum learning rate. The maximum learning rate, which parameterizes the learning rate scheduler, was chosen by carrying out learning rate range tests as described by Smith (2017). To alleviate the sensitivity to noise that this approach has, we used 8 repeats and Kalman smoothing to make the selection more robust. All learning rate curves and chosen learning rates were also manually inspected, and a small number of automatically chosen learning rates were manually overridden where the selection algorithm had failed to correctly interpret the learning rate sweep curve.

Batch and epochs. A batch size of 128 was used. The number of epochs was set to 512; however, this can be less if early stopping is triggered (but always greater than 256). Evaluations were run at the end of every epoch.

Early stopping. An early-stopping strategy was used to reduce wasted compute. After at least 1/2 of the total training steps have been completed (when the learning rate is decreasing), early stopping is triggered if the smoothed validation loss (exponential moving average with smoothing factor 0.2) has not improved over 12 evaluations. Post-training, training and validation loss curves were inspected to ensure that early stopping was not triggered prematurely.

F.1 EDGE SELECTION ALGORITHM RUNTIME

Calculating the edges via the dynamic programming procedure (Algorithm 1) for the main results, with 128 intervals, takes on the order of tens of seconds for the 14 real-world datasets. Figure 25 shows the runtime for the 14 datasets across 4 bin counts, 32, 64, 128, 256.

F.2 COMPUTE RESOURCES

Model training was carried out on 10 Nvidia A40 GPUs from an internal cluster and took ~320 GPU-days. Model evaluation was run on a workstation with GPU, CPU and RAM specifications: Nvidia RTX 4090 GPU, AMD Ryzen Threadripper PRO 5975WX and 256 GiB RAM. Evaluation took ~35 hours.

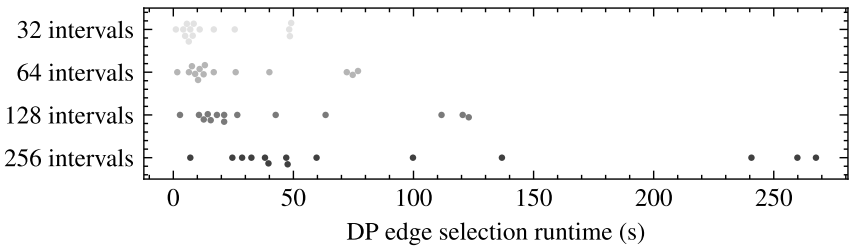


Figure 25: Seconds taken to calculate edges (Algorithm 1) for the 14 real-world datasets across 4 interval counts. Variability is due to differences in the dataset sizes and the number of unique inter-event times within each dataset. All runs used a candidate edge set of size either 15k or 10% of the training set, whichever was less.