A On the relationship between the applied loss and the ELBO

In the following section, we investigate the relationship between our applied loss function and the ELBO to the unknown data distribution derived for diffusion models. The ELBO of diffusion models [18] is given as follows:

$$\mathcal{L}_{ELBO} = \mathbb{E}_q \Big[\underbrace{D_{KL}\left(q(\boldsymbol{t}^{(N)} \mid \boldsymbol{t}^{(0)}) \parallel p(\boldsymbol{t}^{(N)})\right)}_{\mathcal{L}_N} - \underbrace{\log p_\theta(\boldsymbol{t}^{(0)} \mid \boldsymbol{t}^{(1)})}_{\mathcal{L}_0} + \sum_{n=2}^{N} \underbrace{D_{KL}\left(q(\boldsymbol{t}^{(n-1)} \mid \boldsymbol{t}^{(0)}, \boldsymbol{t}^{(n)}) \parallel p_\theta(\boldsymbol{t}^{(n-1)} \mid \boldsymbol{t}^{(n)})\right)}_{\mathcal{L}_n} \Big]. \tag{6}$$

Additionally, the Janossi density [8] of an event sequence $\boldsymbol{t}$ on $[0, T]$ allows us to represent each element of the ELBO in terms of our derived inhomogeneous (approximate) posterior intensities (Section 3.2 and Figure 2) as follows:

$$p(\boldsymbol{t}) = \exp\left(-\int_0^T \lambda(t)\,\mathrm{d}t\right) \prod_{t_i \in \boldsymbol{t}} \lambda(t_i). \tag{7}$$

$\boldsymbol{\mathcal{L}_N}$: $\mathcal{L}_N$ is constant as the intensity λ_{HPP} defining $q(\boldsymbol{t}^{(N)} \mid \boldsymbol{t}^{(0)})$ and $p(\boldsymbol{t}^{(N)})$ has no learnable parameters.

$\boldsymbol{\mathcal{L}_0}$: We directly train our model to optimize this likelihood term as described in Section 3.3.

$\boldsymbol{\mathcal{L}_n}$: The KL divergence between two densities is defined as:

$$D_{KL}(q \parallel p_\theta) = \mathbb{E}_q \left[\log(q(\boldsymbol{t}^{(n-1)} \mid \boldsymbol{t}^{(0)}, \boldsymbol{t}^{(n)})) - \log(p_\theta(\boldsymbol{t}^{(n-1)} \mid \boldsymbol{t}^{(n)}))\right], \tag{8}$$

where only the right-hand side relies on θ, letting us minimize the KL divergence by maximizing the expectation over the log-likelihood $\log(p_\theta(\boldsymbol{t}^{(n-1)} \mid \boldsymbol{t}^{(n)}))$.

To add some additional context to the KL divergence in $\mathcal{L}_n$ and similar to the derivation of the posterior in Section 3.2, we will further distinguish three cases:

1. **Case B & E:** $q(\boldsymbol{t}^{(n-1)} \mid \boldsymbol{t}^{(0)}, \boldsymbol{t}^{(n)})$ and $p_\theta(\boldsymbol{t}^{(n-1)} \mid \boldsymbol{t}^{(n)})$ are defined by Bernoulli distributions over each element of $\boldsymbol{t}^{(n)}$. By definition the cross-entropy $H(q, p)$ of the distribution p relative to q is given by $H(q, p) = H(q) + D_{KL}(q \parallel p)$, where $H(q)$ is the entropy and D_{KL} the KL divergence. We can see that minimizing the (binary) cross-entropy is equivalent to minimizing the KL divergence, as the entropy $H(q)$ is constant for the data distribution.
2. **Case C:** Minimizing this KL divergence by maximizing the $\mathbb{E}_q\left[\log(p_\theta(\boldsymbol{t}^{(n-1)} \mid \boldsymbol{t}^{(n)}))\right]$ is proposed in Section 3.2 to learn $\lambda_\theta^{(A \cup C)}(t)$. Note that $q(\boldsymbol{t}^{(n-1)} \mid \boldsymbol{t}^{(0)}, \boldsymbol{t}^{(n)})$ is sampled from by independently thinning $\boldsymbol{t}^{(0)} \setminus \boldsymbol{t}^{(n)}$. Consequently, by minimizing the NLL of our intensity $\lambda_\theta^{(A \cup C)}(t)$ with regards to $\boldsymbol{t}^{(0)} \setminus \boldsymbol{t}^{(n)}$, we optimize the expectation in closed form.
3. **Case D:** Our parametrization uses the same intensity function for $q(\boldsymbol{t}^{(n-1)} \mid \boldsymbol{t}^{(0)}, \boldsymbol{t}^{(n)})$ and $p_\theta(\boldsymbol{t}^{(n-1)} \mid \boldsymbol{t}^{(n)})$, which does not rely on any learned parameters.

B Derivations

B.1 Direct forward sampling

Proof. We first repeat Equation 2:

$$\lambda_n(t) = \alpha_n \lambda_{n-1}(t) + (1 - \alpha_n)\lambda_{\text{HPP}}. \tag{2}$$

Then for the first step we can write:

$$\begin{aligned} \lambda_1(t) &= \alpha_1 \lambda_0(t) + (1 - \alpha_1)\lambda_{\text{HPP}} \\ &= \left(\prod_{j=1}^{n=1} \alpha_j\right) \lambda_0 + \left(1 - \prod_{j=1}^{n=1} \alpha_j\right) \lambda_{\text{HPP}}. \end{aligned}$$

Assuming Equation 3 holds for step $n-1$:

$$\lambda_{n-1}(t) = \left(\prod_{j=1}^{n-1} \alpha_j\right) \lambda_0(t) + \left(1 - \prod_{j=1}^{n-1} \alpha_j\right) \lambda_{\text{HPP}},$$

we can write for step n:

$$\begin{aligned}
\lambda_n(t) &= \alpha_n \lambda_{n-1}(t) + (1-\alpha_n)\lambda_{\text{HPP}} \\
&= \alpha_n \left(\left(\prod_{j=1}^{n-1} \alpha_j\right) \lambda_0(t) + \left(1 - \prod_{j=1}^{n-1} \alpha_j\right) \lambda_{\text{HPP}} \right) + (1-\alpha_n)\lambda_{\text{HPP}} \\
&= \left(\prod_{j=1}^{n} \alpha_j\right) \lambda_0(t) + \left(\alpha_n - \prod_{j=1}^{n} \alpha_j\right) \lambda_{\text{HPP}} + (1-\alpha_n)\lambda_{\text{HPP}} \\
&= \left(\prod_{j=1}^{n} \alpha_j\right) \lambda_0(t) + \left(1 - \prod_{j=1}^{n} \alpha_j\right) \lambda_{\text{HPP}} \\
&= \bar{\alpha}_n \lambda_0(t) + (1-\bar{\alpha}_n)\lambda_{\text{HPP}},
\end{aligned}$$

which completes the proof by induction. □

B.2 Conditional distribution of Poisson variables

Proposition. *Given two independent random variables $X_1 \sim Poisson(\lambda_1)$, $X_2 \sim Poisson(\lambda_2)$, $X_1 \mid X_1 + X_2 = k$ is Binomial distributed, i.e., $X_1 \mid X_1 + X_2 = k \sim Binomial(x_1; k, \frac{\lambda_1}{\lambda_1+\lambda_1})$.*

Proof. The Poisson distributed random variables X_1 and X_2 have the following joint probability mass function:

$$P(X_1 = x_1, X_2 = x_2) = e^{-(\lambda_1)} \frac{\lambda_1^{x_1}}{x_1!} e^{-\lambda_2} \frac{\lambda_2^{x_2}}{x_2!}, \tag{9}$$

which further defines the joint probability mass function of $P(X_1 = x_1, X_2 = x_2, Y = k)$ if $x_1 + x_2 = k$. Additionally, it is well know that $Y = X_1 + X_2$ is a Poisson random variable with intensity $\lambda_1 + \lambda_2$ and therefore $P(Y = k) = e^{-(\lambda_1+\lambda_2)} \frac{(\lambda_1+\lambda_2)^k}{k!}$. Then $P(X_1 = x_1 \mid Y = k)$ is given by the following:

$$P(X_1 = x_1 \mid Y = k) = \frac{P(X_1 = x_1, X_2 = x_2, Y = k)}{P(Y = k)} \text{ if } x_1 + x_2 = k, \tag{10}$$

$$= \frac{e^{-\lambda_1} \frac{\lambda_1^{x_1}}{x_1!} e^{-\lambda_2} \frac{\lambda_2^{x_2}}{x_2!}}{e^{-(\lambda_1+\lambda_2)} \frac{(\lambda_1+\lambda_2)^k}{k!}} \text{ if } x_1 + x_2 = k, \tag{11}$$

$$= \frac{k!}{x_1! x_2!} \frac{\lambda_1^{x_1} \lambda_2^{x_2}}{(\lambda_1+\lambda_2)^k} \text{ if } x_1 + x_2 = k, \tag{12}$$

$$= \frac{k!}{x_1!(k-x_1)!} \frac{\lambda_1^{x_1}}{(\lambda_1+\lambda_2)^{x_1}} \frac{\lambda_2^{k-x_1}}{(\lambda_1+\lambda_2)^{k-x_1}} \text{ if } x_1 + x_2 = k, \tag{13}$$

$$= \frac{k!}{x_1!(k-x_1)!} \left(\frac{\lambda_1}{(\lambda_1+\lambda_2)}\right)^{x_1} \left(1 - \frac{\lambda_1}{(\lambda_1+\lambda_2)}\right)^{(k-x_1)} \text{ if } x_1 + x_2 = k, \tag{14}$$

where we have leveraged $x_2 = k - x_1$ and $\frac{\lambda_2}{(\lambda_1+\lambda_2)} = 1 - \frac{\lambda_1}{(\lambda_1+\lambda_2)}$. As we have shown $P(X_1 = x_1 \mid Y = k)$ follows the Binomial distribution with $p = \frac{\lambda_1}{(\lambda_1+\lambda_2)}$. □

Table 5: Statistics for the synthetic datasets.

	# Sequences	T	Average sequence length	τ
Hawkes1	1000	100	95.4	1.01 ± 2.38
Hawkes2	1000	100	97.2	0.98 ± 2.56
SC	1000	100	100.2	0.99 ± 0.71
IPP	1000	100	100.3	0.99 ± 2.22
RP	1000	100	109.2	0.83 ± 2.76
MRP	1000	100	98.0	0.98 ± 1.83

Table 6: Statistics for the real-world datasets.

	# Sequences	T	Unit of time	Average sequence length	τ	ΔT
PUBG	3001	30	minutes	76.5	0.41 ± 0.56	5
Reddit-C	1356	24	hours	295.7	0.07 ± 0.28	4
Reddit-S	1094	24	hours	1129.0	0.02 ± 0.03	4
Taxi	182	24	hours	98.4	0.24 ± 0.40	4
Twitter	2019	24	hours	14.9	1.26 ± 2.80	4
Yelp1	319	24	hours	30.5	0.77 ± 1.10	4
Yelp2	319	24	hours	55.2	0.43 ± 0.96	4

C Datasets

Synthetic datasets. The six synthethic dataset were sampled by Shchur et al. [42] following the procedure in Section 4.1 of Omi et al. [37] and consist of 1000 sequences on the interval $[0, 100]$.

Real-world datasets. The seven real-world datasets were proposed by Shchur et al. [42] and consist of PUBG, Reddit-Comments, Reddit-Submissions, Taxi, Twitter, Yelp1, and Yelp2. The event sequences of **PUBG** represent the death of players in a game of Player Unknown's Battleground (PUBG). The event sequences of **Reddit-Comments** represent the comments on the askscience subreddit within 24 hours after opening the discussion in the period from 01.01.2018 until 31.12.2019. The event sequences of **Reddit-Submissions** represent the discussion submissions on the politics subreddit within a day in the period from 01.01.2017 until 31.12.2019. The event sequences of **Taxi** represent taxi pick-ups in the south of Manhattan, New York. The event sequences of **Twitter** represent tweets by user 25073877. The event sequences of **Yelp1** represent check-ins for the McCarran International Airport recorded for 27 users in 2018. The event sequences of **Yelp2** represent check-ins for all businesses in the city of Mississauga recorded for 27 users in 2018.

We report summary statistics on the datasets in Table 5 and 6. Lately, the validity of some of the widely used real-world benchmark datasets was criticized [4]. In one-step-ahead prediction tasks with teacher forcing, very simple architectures achieved similar results to some of the more advanced ones. However, this seems to be more of a problem of the task than the datasets. In our work, we consider different tasks (density estimation and long-term forecasting) and metrics and have found significant empirical differences between the baselines on these datasets.

D Experimental set-up

All models but the transformer baseline were trained on an Intel Xeon E5-2630 v4 @ 2.20 GHz CPU with 256GB RAM and an NVIDIA GeForce GTX 1080 Ti. Given its RAM requirement, the transformer baseline had to be trained with batch size 32 on an NVIDIA A100-PCIE-40GB for the Reddit-C and Reddit-S datasets.

Hyperparameter tuning. has been applied to all models. The hyperparameter tuning was done on the MMD loss between 1000 samples from the model and the validation set. We use a hidden dimension of 32 for all models. Further, we have tuned the learning rate in $\{0.01, 0.001\}$ for all models, the number of mixture components in $\{8, 16\}$ for ADD-THIN, *RNN* and *Transformer*, the number of knots in $\{10, 20\}$ for *TriTPP* and the number of attention layers in $\{2, 3\}$ for the transformer baseline. The values of all other baseline hyperparameters were set to the recommended

values given by the authors. Further, the *GD* baseline has been trained with a batch size of 16, as recommended by the authors. For the forecasting task, we apply the optimal hyperparameters from the density estimation experiment.

Early-stopping. Each model has been trained for up to 5000 epochs with early stopping on the MMD metric on the validation set for the density estimation task and on the Wasserstein distance metric on the validation set for the forecasting task.

E Additional results

E.1 Density estimation results with standard deviations

Table 7: **Synthetic data**: MMD (↓) between the TPP distribution of sampled sequences and hold-out test set.

	Hawkes1	Hawkes2	SC	IPP	RP	MRP
RNN	0.02±0.003	0.01±0.002	0.08±0.053	0.05±0.009	0.01±0.001	0.03±0.005
Transformer	0.03±0.011	0.04±0.017	0.19±0.006	0.10±0.034	0.02±0.007	0.19±0.048
GD	0.06±0.004	0.06±0.002	0.13±0.004	0.08±0.002	0.05±0.002	0.14±0.008
TriTPP	0.03±0.002	0.04±0.001	0.23±0.003	0.04±0.003	0.02±0.002	0.05±0.004
ADD-THIN (Ours)	0.02±0.004	0.02±0.002	0.19±0.013	0.03±0.006	0.02±0.001	0.10±0.030

Table 8: **Real-world data**: MMD (↓) between the TPP distribution of sampled sequences and hold-out test set.

	PUBG	Reddit-C	Reddit-S	Taxi	Twitter	Yelp1	Yelp2
RNN	0.04±0.005	0.01±0.002	0.02±0.003	0.04±0.001	0.03±0.003	0.07±0.005	0.03±0.001
Transformer	0.06±0.014	0.05±0.025	0.09±0.06	0.09±0.014	0.08±0.02	0.12±0.026	0.14±0.048
GD	0.11±0.023	0.03±0.001	0.03±0.001	0.1±0.002	0.15±0.011	0.12±0.01	0.1±0.001
TriTPP	0.06±0.001	0.09±0.002	0.12±0.003	0.07±0.007	0.04±0.002	0.06±0.005	0.06±0.004
ADD-THIN (Ours)	0.03±0.015	0.01±0.005	0.02±0.001	0.04±0.006	0.04±0.006	0.08±0.01	0.04±0.005

Table 9: **Synthetic data**: Wasserstein distance (↓) between the distribution of the number of events of sampled sequences and hold-out test set.

	Hawkes1	Hawkes2	SC	IPP	RP	MRP
RNN	0.03±0.007	0.01±0.002	0.00±0.003	0.02±0.006	0.02±0.002	0.01±0.004
Transformer	0.06±0.017	0.04±0.01	0.06±0.008	0.07±0.035	0.04±0.005	0.11±0.048
GD	0.16±0.016	0.13±0.012	0.5±0.025	0.42±0.009	0.28±0.039	0.5±0.035
TriTPP	0.03±0.003	0.03±0.001	0.01±0.0	0.01±0.001	0.02±0.003	0.03±0.001
ADD-THIN (Ours)	0.04±0.009	0.02±0.006	0.08±0.018	0.01±0.003	0.02±0.001	0.04±0.006

Table 10: **Real-world data**: Wasserstein distance (↓) between the distribution of the number of events of sampled sequences and hold-out test set.

	PUBG	Reddit-C	Reddit-S	Taxi	Twitter	Yelp1	Yelp2
RNN	0.02±0.004	0.01±0.004	0.05±0.013	0.02±0.002	0.01±0.001	0.04±0.004	0.02±0.002
Transformer	0.04±0.013	0.08±0.028	0.11±0.032	0.13±0.073	0.05±0.021	0.11±0.03	0.21±0.077
GD	0.54±0.054	0.02±0.004	0.16±0.013	0.33±0.007	0.07±0.062	0.26±0.012	0.25±0.007
TriTPP	0.03±0.003	0.09±0.001	0.09±0.001	0.04±0.001	0.01±0.001	0.03±0.006	0.04±0.002
ADD-THIN (Ours)	0.02±0.009	0.03±0.007	0.04±0.002	0.03±0.007	0.01±0.004	0.04±0.006	0.02±0.006

E.2 Forecasting results with standard deviations

Table 11: Wasserstein distance (↓) between forecasted event sequence and ground truth reported for 50 random forecast windows on the test set.

	PUBG	Reddit-C	Reddit-S	Taxi	Twitter	Yelp1	Yelp2
Average Seq. Length	76.5	295.7	1129.0	98.4	14.9	30.5	55.2
RNN	6.15±2.53	35.22±4.02	39.23±2.06	4.14±0.25	2.04±0.08	1.28±0.03	2.21±0.06
Transformer	2.45±0.21	38.77±10.68	27.52±5.24	3.12±0.1	2.09±0.07	1.29±0.1	2.64±0.24
GD	5.44±0.2	44.72±1.77	64.25±4.45	4.32±0.3	2.16±0.23	1.52±0.15	4.25±0.46
ADD-THIN (Ours)	2.03±0.01	17.18±1.18	21.32±0.42	2.42±0.03	1.48±0.03	1.0±0.02	1.54±0.04

Table 12: Count MAPE $\times 100\%$ (↓) between forecasted event sequences and ground truth reported for 50 random forecast windows on the test set.

	PUBG	Reddit-C	Reddit-S	Taxi	Twitter	Yelp1	Yelp2
Average Seq. Length	76.5	295.7	1129.0	98.4	14.9	30.5	55.2
RNN	1.72±0.65	5.47±0.92	0.68±0.07	0.54±0.02	0.95±0.08	0.59±0.02	0.72±0.03
Transformer	0.65±0.11	7.38±2.51	0.55±0.14	0.46±0.04	1.18±0.09	0.63±0.08	0.99±0.11
GD	1.66±0.06	10.49±0.42	1.33±0.12	0.71±0.05	1.43±0.2	0.78±0.1	1.65±0.2
ADD-THIN (Ours)	0.45±0.005	1.07±0.19	0.38±0.02	0.37±0.02	0.69±0.03	0.45±0.02	0.5±0.03

E.3 Sampling runtimes

We compare sampling runtimes on an NVIDIA GTX 1080 Ti across the different models in Figure 5. ADD-THIN maintains near-constant runtimes by refining the entire sequence in parallel. The autoregressive baselines *RNN* and *Transformer* show increasing runtimes, with longer sequences surpassing ADD-THIN's runtime. *TriTPP* is a highly optimized baseline computing the autoregressive interactions between event times in parallel by leveraging triangular maps, resulting in the fastest runtimes. Lastly, *GD* is autoregressive in event time and gradually refines each event time over 100 diffusion steps, leading to the worst runtimes.

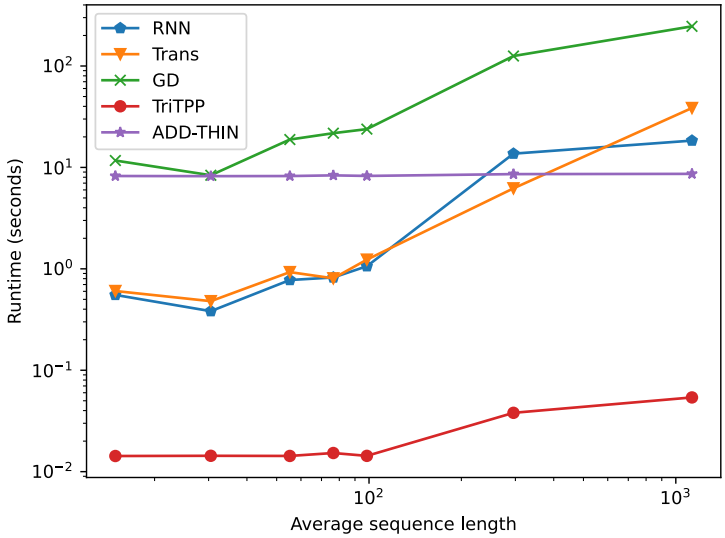

Figure 5: Sampling runtime for a batch of 100 event sequences averaged over 100 runs. We report the trained model's sampling times for the real-world datasets with different sequence lengths (from left to right: Twitter, Yelp 1, Yelp 2, PUBG, Taxi, Reddit-C, Reddit-A).