A Derivation of the Evidence Lower Bound

We provide the detailed steps to derive the evidence lower bound of the proposed Meta TPP model shown in Equation (7). Here, Equation (8) to (9) holds with Jensen's inequality.

$$\begin{aligned}\log p_\theta(\tau_l \,|\, \tau_{l-k:l-1}, \mathcal{C}_l) &= \log \int p_\theta(\tau_l \,|\, \tau_{l-k:l-1}, z) p_\theta(z \,|\, \mathcal{C}_l) dz \\ &= \log \int p_\theta(z|\mathcal{C}_L) \frac{p_\theta(z|\mathcal{C}_l)}{p_\theta(z|\mathcal{C}_L)} p_\theta(\tau_l|\tau_{l-k:l-1}, z) dz \\ &\geq \int p_\theta(z|\mathcal{C}_L) \log \frac{p_\theta(z|\mathcal{C}_l)}{p_\theta(z|\mathcal{C}_L)} p_\theta(\tau_l|\tau_{l-k:l-1}, z) dz \\ &= \mathbb{E}_{z\sim p_\theta(z \,|\, \mathcal{C}_L)} \left[\log p_\theta(\tau_l \,|\, \tau_{l-k:l-1}, z)\right] - KL(p_\theta(z \,|\, \mathcal{C}_L) \,|\, p_\theta(z \,|\, \mathcal{C}_l)).\end{aligned}$$

B Role of Global Latent Feature

For the global latent feature z to be task-specific, it has to be distinct for different sequences but similar throughout different event times $l \in [1, L-1]$ within the same sequence as mentioned in Section 3.2. It is natural for the global features to be distinct by sequence but we need further guidance to make the global feature shared across all the event times in a sequence. In fact, due to the permutation invariance constraint (implemented in average-pooling), the global latent feature z cannot be very different at different event time: adding some addition context features r_i will not change G as well as z much.

For the latent variable z, additional guidance is provided, which is clearer with the objective of the variational inference. Recall that the objective of the variational inference in Equation (6) is provided as,

$$\arg\max_\theta E_{z\sim p_\theta(z \,|\, \mathcal{C}_L)} \log p_\theta(\tau_l \,|\, \tau_{l-k:l-1}, z) - KL(p_\theta(z \,|\, \mathcal{C}_L) \,||\, p_\theta(z \,|\, \mathcal{C}_l)).$$

Here, regardless of the index of the target l, it always minimizes the KL divergence between $p_\theta(z \,|\, \mathcal{C}_L)$ and $p_\theta(z \,|\, \mathcal{C}_l)$ where L is the length of a sequence. So, ideally, the latent variable $z \sim p_\theta(z \,|\, \mathcal{C}_l)$ should capture the same information as $z \sim p_\theta(z \,|\, \mathcal{C}_L)$. It implies regardless of the index of the target l, the latent variable z asymptotically captures the global feature of the whole sequence, which is equivalent to $z \sim p_\theta(z \,|\, \mathcal{C}_L)$. Hence, the resulting $p_\theta(z \,|\, \mathcal{C}_l)$ captures the global and task-specific patterns, which ideally is similar to $p_\theta(z \,|\, \mathcal{C}_L)$. As a result, the global latent feature is guided to be distinct for different sequences but similar throughout different event time $l \in [1, L-1]$ within the same sequence.

It is also worth mentioning that its role is quite different from the target input $\tau_{l-k+1:l}$ which is another input for the decoder. Consider two events that are far apart from each other. Due to distinctive local patterns, their target inputs can be quite different from each other. On the other hand, the global features will not be that different as they are the average of all the context features at each event time, plus guided by the KL divergence. Hence, the global feature provides "overall" patterns of a task whereas a target input provides local patterns to the decoder.

Back to our original goal: treating each sequence as a realization of a distinct stochastic process, we use the global latent feature that is distinct by each sequence to provide a task-specific information which is shared regardless of different event time step l. It is neither implicitly nor explicitly considered in the supervised learning case. In supervised learning, each event time step at each sequence is treated equally from which patterns for only one stochastic process is learned.

In the Table 5, we compare the THP$^+$ baseline and Meta TPP on Sinusoidal, Uber and NYC Taxi datasets to demonstrate the effectiveness of the global latent feature. The decoder of the Meta TPP takes the global latent feature z (from the permutation invariance constraint) as an input, in addition to the target input feature r_l that the decoder of the THP$^+$ baseline takes as input. The result shows that the global latent feature generally helps to improve both RMSE and NLL performance.

Methods	Sinusoidal		Uber		NYC Taxi	
	RMSE	NLL	RMSE	NLL	RMSE	NLL
THP$^+$	1.72	0.84	90.25	3.63	10.31	**2.00**
Meta TPP	**1.48**	**0.61**	**63.35**	**3.25**	**10.04**	2.33

Table 5: Comparison between THP$^+$ baseline and Meta TPP.

C Computation of Evaluation Metrics

Unlike Equation (7) in the main paper where the ELBO is computed using samples from $p_\theta(z \,|\, \mathcal{C}_L)$, in inference, we do not have access to $z \sim p_\theta(z \,|\, \mathcal{C}_L)$. But, as $p_\theta(z \,|\, \mathcal{C}_l)$ is trained to be similar to $p_\theta(z \,|\, \mathcal{C}_L)$ through $KL(p_\theta(z \,|\, \mathcal{C}_L) \,|\, p_\theta(z \,|\, \mathcal{C}_l)$, we use samples $z \sim p_\theta(z \,|\, \mathcal{C}_l)$. As specified in Appendix I, we use 256 samples to have good enough approximation.

- NLL – We approximate a log-likelihood of the next event time τ_{l+1} using Monte-Carlo approximation as,

$$\log p_\theta(\tau_{l+1} \,|\, \tau_{l-k+1:l}, \mathcal{C}_l) = \log \int p_\theta(\tau_{l+1} \,|\, \tau_{l-k+1:l}, z) p_\theta(z \,|\, \mathcal{C}_l) dz \tag{11}$$

$$\approx \log \frac{1}{M} \sum_{m=1}^{M} p_\theta(\tau_{l+1} \,|\, \tau_{l-k+1:l}, z_m) \tag{12}$$

  where M is the number of samples from $p_\theta(z \,|\, \mathcal{C}_l)$.

- RMSE – We use a mixture of log-normal distributions to model $p_\theta(\tau_{l+1} \,|\, \tau_{l-k+1:l}, z)$. Formally, for $l \in [1, L-1]$, $\tau_{l+1} \sim MixLogNorm(\boldsymbol{\mu}_{l+1}, \boldsymbol{\sigma}_{l+1}, \boldsymbol{\omega}_{l+1})$ where $\boldsymbol{\mu}_{l+1}$ are the mixture means, $\boldsymbol{\sigma}_{l+1}$ are the standard deviations, and $\boldsymbol{\omega}_{l+1}$ are the mixture weights. The parameters are the outputs of the decoder given a latent sample z. Knowing this, we can analytically compute the expected event time for a latent sample z with K mixture components as,

$$E_{\tau_{l+1} \sim p_\theta(\tau_{l+1} \,|\, \tau_{l-k+1:l}, z)}[\tau_{l+1}] = \sum_{k=1}^{K} \omega_{l+1,k} \exp\left(\mu_{l+1,k} + \frac{1}{2}\sigma_{l+1,k}^2\right).$$

  Note that since this expectation is over $p_\theta(\tau_{l+1} \,|\, \tau_{l-k+1:l}, z)$ where z is one sample from the posterior, we need to take another expectation over the posterior as follows,

$$E_{\tau_{l+1} \sim p_\theta(\tau_{l+1} \,|\, \tau_{l-k+1:l}, \mathcal{C}_l)}[\tau_{l+1}] = E_{z \sim p_\theta(z \,|\, \mathcal{C}_l)} E_{\tau_{l+1} \sim p_\theta(\tau_{l+1} \,|\, \tau_{l-k+1:l}, z)}[\tau_{l+1}] \tag{13}$$

$$= E_{z \sim p_\theta(z \,|\, \mathcal{C}_l)} \sum_{k=1}^{K} \omega_{l+1,k} \exp\left(\mu_{l+1,k} + \frac{1}{2}\sigma_{l+1,k}^2\right) \tag{14}$$

$$\approx \frac{1}{M} \sum_{m=1}^{M} \sum_{k=1}^{K} \omega_{l+1,k} \exp\left(\mu_{l+1,k} + \frac{1}{2}\sigma_{l+1,k}^2\right) \tag{15}$$

  where M is the number of samples from $p_\theta(z \,|\, \mathcal{C}_l)$.

- Accuracy – We obtain class predictions by taking argmax over the probability distribution of class labels as follows,

$$\arg\max_{c \in [1, C]} p_\theta(y_{l+1} \,|\, \tau_{l-k+1:l}, y_{l-k+1:l}, \mathcal{C}_l)$$

  where C is the number of marks. The probability distribution of class labels is approximated using MC samples as,

$$p_\theta(y_{l+1} \,|\, \tau_{l-k+1:l}, y_{l-k+1:l}, \mathcal{C}_l) = \int p_\theta(y_{l+1} \,|\, r_l, z) p_\theta(z \,|\, \mathcal{C}_l) dz \tag{16}$$

$$\approx \frac{1}{M} \sum_{m=1}^{M} p_\theta(y_{l+1} \,|\, r_l, z_m) \tag{17}$$

  where M is the number of samples from $p_\theta(z \,|\, \mathcal{C}_l)$.

D Monte-Carlo Approximation vs. Variational Inference

Amortized variational inference (VI) we described in Section 3 is not the only way to approximate the latent variable model. We can also use Monte-Carlo (MC) approximation, which is simpler and does not rely on a proxy like ELBO. It is formulated as,

$$\log \int p_\theta(\tau_{l+1} \,|\, \tau_{l-k+1:l}, z) p_\theta(z \,|\, \mathcal{C}_l) dz \approx \log \frac{1}{N} \sum_{n=1}^{N} p_\theta(\tau_{l+1} \,|\, \tau_{l-k+1:l}, z_n) \tag{18}$$

Note that a sample z_n in Equation (18) is drawn from $p(z_n|\mathcal{C}_l)$, which is different from $z_n \sim p(z_n|\mathcal{C}_L)$ in Equation (7). Foong et al. (2020); Dubois et al. (2020) report that MC approximation generally outperforms variational inference. In variational inference, as a model is trained with $z \sim p_\theta(z \,|\, \mathcal{C}_L)$, the samples $z \sim p_\theta(z \,|\, \mathcal{C}_l)$ in test time are quite different from what the model has used as inputs: although $KL(p_\theta(z \,|\, \mathcal{C}_L) \,|\, p_\theta(z \,|\, \mathcal{C}_l))$ forces $p_\theta(z \,|\, \mathcal{C}_L)$ and $p_\theta(z \,|\, \mathcal{C}_l)$ close to each other, it is hard to make KL-divergence to be zero. Although MC approximation outperforms VI for the proposed Meta TPP, it is not the case for the Attentive TPP as shown in Table 6.

Attention	Latent		Reddit			Uber		NYC Taxi	
	VI	MC	RMSE	NLL	Acc	RMSE	NLL	RMSE	NLL
✗	✓	✗	0.13	**-0.39**	**0.61**	63.35	3.25	**10.04**	2.33
✗	✗	✓	**0.11**	0.16	**0.61**	**37.12**	**3.22**	10.15	**2.00**
✓	✓	✗	**0.11**	0.03	**0.60**	**21.87**	**2.98**	**8.92**	**2.00**
✓	✗	✓	0.13	**-0.05**	**0.60**	22.38	3.18	9.10	2.01

Table 6: Comparison of the variants of Meta TPPs

We conjecture it based on MC approximation better sharing role with the cross-attention path when compared to the VI approximation. More specifically, cross-attention forces a model to have similar features for repeating local history patterns. As it focuses on extracting features from the previous history, which is similar to what $z \sim p_\theta(z \,|\, \mathcal{C}_l)$ contains, the latent and attentive path share a role in MC approximation. On the other hand, as the model is trained on the global latent feature $z \sim p_\theta(z \,|\, \mathcal{C}_L)$ in VI, without focusing too much on the previous history due to the cross-attention, it may be able to utilize more diverse features. It would be an interesting future work to investigate the theoretical relationship between approximation methods and variants of Meta TPP.

E Naïve Baseline

Sometimes naïve baselines can be stronger baselines than more sophisticated ones. To investigate if that is the case for TPPs, we implement a naïve baseline that makes predictions based on median inter-event interval: $\hat{\tau}\_l + 1 = \tau\_l + \Delta\tau\_median, l$ where $\Delta\tau\_median, l$ is a median of the inter-event interval up to l-th event. We boostrap for 200 times on the test set to obtain the mean of RMSE metrics as with how we obtain the numbers for the other methods (NLLs are not available for the naïve baseline). In Table 7, the performance of the naïve baseline is surprisingly good for some cases. For instance, it is better than the intensity-free on Wiki and NYC Taxi datasets. It is, however, much worse than THP$^+$ and the proposed Attentive TPP on all the datasets.

Methods	Stack Overflow	Mooc	Reddit	Wiki	Sinusoidal	Uber	NYC Taxi
Naïve baseline	161.21	0.79	0.38	0.21	4.61	107.91	24.58
Intensity-free	3.64	0.31	0.18	0.60	1.29	51.23	46.59
Neural flow	–	0.47	0.32	0.56	**1.13**	–	–
THP$^+$	1.68	0.18	0.26	0.17	1.72	90.25	10.31
Attentive TPP	**1.15**	**0.16**	**0.11**	**0.15**	1.45	**22.11**	**8.92**

Table 7: Comparison of Naïve baseline and other methods

F Effect of Model Size

Although we have demonstrated that the improvement in performance does not come from the size of a model through Table 4 on the Reddit dataset, we provide more evidence on the rest of the datasets as below.

Methods	# Params	Sinusoidal		Uber		NYC Taxi	
		RMSE	NLL	RMSE	NLL	RMSE	NLL
THP^{+}	50K	1.72 (0.10)	0.84 (0.02)	90.25 (4.53)	3.63 (0.03)	10.31 (0.47)	2.00 (0.01)
	100K	1.84 (0.13)	1.04 (0.02)	82.69 (4.56)	3.34 (0.03)	10.16 (0.47)	**1.92 (0.01)**
Attentive TPP	96K	**1.45 (0.11)**	**0.66 (0.02)**	**22.11 (1.94)**	**2.89 (0.04)**	**8.92 (0.42)**	2.00 (0.009)

Table 8: Comparison of different model size on periodic datasets.

Methods	# Params	Stack Overflow		
		RMSE	NLL	Acc
THP^{+}	52K	1.68 (0.16)	3.28 (0.02)	**0.46 (0.004)**
	103K	1.63 (0.06)	2.82 (0.03)	**0.46 (0.004)**
Attentive TPP	99K	**1.15 (0.02)**	**2.64 (0.02)**	**0.46 (0.004)**

Table 9: Comparison of different model size on the Stack Overflow dataset.

Methods	# Params	Mooc		
		RMSE	NLL	Acc
THP^{+}	56K	0.18 (0.005)	0.13 (0.02)	0.38 (0.004)
	113K	0.22 (0.007)	0.05 (0.03)	**0.39 (0.004)**
Attentive TPP	108K	**0.16 (0.004)**	**-0.72 (0.02)**	0.36 (0.003)

Table 10: Comparison of different model size on the Mooc dataset.

Methods	# Params	Wiki		
		RMSE	NLL	Acc
THP^{+}	577K	0.17 (0.02)	**6.25 (0.39)**	0.23 (0.03)
	1153K	0.16 (0.01)	6.47 (0.40)	0.21 (0.02)
Attentive TPP	1149K	**0.15 (0.01)**	**6.25 (0.38)**	**0.25 (0.03)**

Table 11: Comparison of different model size on the Wiki dataset.

In the table above, we observe that in many cases, smaller models perform better than larger models : on Sinusoidal, Mooc, and Wiki. Although it is sometimes true that larger models perform better than their smaller counterparts, they are still significantly worse than our proposed Attentive TPP. Note that we conducted exactly the same grid search for hyperparameter tuning for the larger models. The results empirically demonstrate that the improvement does not necessarily come from the size of a model but from right inductive biases.

Lastly, please note that all the experiment results we have reported in Table 1-4 and in rebuttal are on the test sets. We believe the RMSE, NLL, and Accuracy on test sets are good metrics to compare the generalization performance of different models. Given that our proposed method outperforms all the baselines, we think our experiments empirically demonstrate the robustness of our method in terms of generalization.

G Extension to Marked TPPs

We extended the proposed method to the marked cases by adding class prediction branch following the intensity-free TPP (Shchur et al., 2020). Suppose a mark at $l+1$-th event is denoted as y_{l+1}.

Datasets	# of Seq.	# of Events	Max Seq. Length	# of Marks
Stack Overflow	6,633	480,414	736	22
Mooc	7.047	389,407	200	97
Reddit	10,000	532,026	100	984
Wiki	1,000	138,705	250	7,628
Sinusoidal	1,000	107,454	200	1
Uber	791	701,579	2,977	1
NYC Taxi	1,000	1,141,379	1,958	1

Table 12: Statistics of the datasets.

For the proposed Meta TPP, we compute the log-likelihood of the mark as,

$$\log p_\theta(y_{l+1} \,|\, \tau_{l-k+1:l}, y_{l-k+1:l}, \mathcal{C}_l) = \log \int p_\theta(y_{l+1} \,|\, \tau_{l-k+1:l}, y_{l-k+1:l}, z) p_\theta(z \,|\, \mathcal{C}_l) dz.$$

Note that $\mathcal{C}_l$ includes both event times and corresponding labels. For implementation, we added one fully connected layer that takes as input the same features for the decoder (that predicts the next event time), and outputs the logits for classification. A class prediction is made by taking argmax over the probability distribution which is approximated using Monte-Carlo samples as,

$$p_\theta(y_{l+1} \,|\, \tau_{l-k+1:l}, y_{l-k+1:l}, \mathcal{C}_l) \approx \frac{1}{M} \sum_{m=1}^{M} p_\theta(y_{l+1} \,|\, r_l, z_m)$$

Note that inputs $\tau_{l-k+1:l}$ and $y_{l-k+1:l}$ are encoded to r_l. The class predictions to compute the accuracies reported throughout the experiments are made from this.

H Description of Datasets

We use 4 popular benchmark datasets: Stack Overflow, Mooc, Reddit, and Wiki, and 3 newly processed datasets: Sinusoidal wave, Uber and NYC Taxi. The statistics are provided in Table 12. To split the data into train, validation, and test, we follow the splits made in the previous works such as Shchur et al. (2020), and Yang et al. (2022). More specifically, we use 60%, 20%, and 20% split for train, validation, and test, respectively, for all the datasets following Shchur et al. (2020) except for Stack Overflow. For Stack Overflow, we follow the split made by Yang et al. (2022) and Du et al. (2016) where 4,777, 530, and 1,326 samples are assigned for train, validation, and test, respectively. For more detailed descriptions of the popular benchmark datasets, please refer to the original papers as described below or Section E.2 of Shchur et al. (2020).

H.1 Benchmark Datasets

Stack Overflow. It was first processed in (Du et al., 2016). We use the first folder of the dataset following (Shchur et al., 2020; Yang et al., 2022).

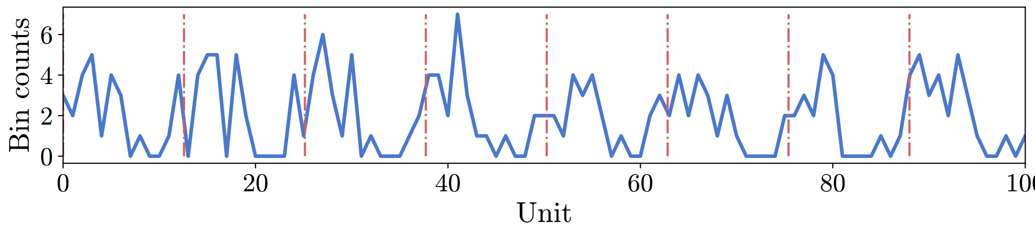
Mooc. It consists of 7,047 sequences, each of which contains of action times an individual user of an online Mooc course. There are 98 categories.

Reddit. It consists of 10,000 sequences from the most active users with marks being the sub-reddit categories of each sequence.

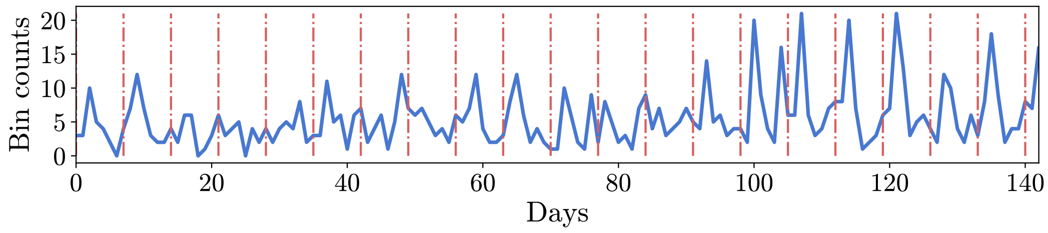
Wiki. It consists of 1,000 sequences from the most edited Wikipedia pages (for a month period) with marks being users who made at least 5 changes.

H.2 Proposed Datasets

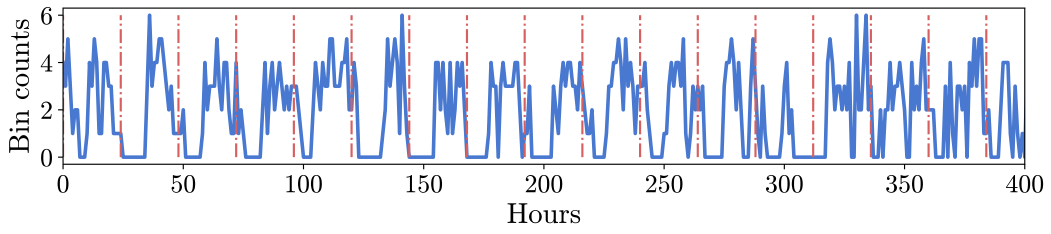
Sinusoidal wave. We generate the Sinusoidal wave using a sine function with a periodicity of 4π and the domain of $[0, 32\pi]$. We randomly choose the number of events per sequence in $[20, 200]$ for $1{,}000$ sequences.



(a) An example in Sinusoidal wave dataset.



(b) An example in Uber dataset.



(c) An example in NYC Taxi dataset.

Figure 4: Visualization of examples in the datasets with strong periodicity.

Uber.[2] We generate the Uber dataset using the data from the shared link. Among the data from Januaray, 2015 to June, 2015, we create sequences using *Dispatching-base-num* and *locationID* as keys. We also give a constraint that the mininum and maximum events per sequence being 100 and 3,000, respectively, and drop all the overlapping event times.

NYC Taxi.[3] We generate the NYC Taxi dataset from the NYC Taxi pickup raw data in 2013 shared in the link, which is different from the one proposed in Du et al. (2016), as we do not include any location information. We generate 6 different datasets by splitting the whole data in 2013 for every two months: Jan-Feb, Mar-Apr, May-Jun, Jul-Aug, Sep-Oct, and Nov-Dec. Throughout the experiment, we train models on the training set of Jan-Feb split, and evaluate on the test set of Jan-Feb for in-domin, and other for distribution drift.

H.3 Periodicity

In Section 5.2, we provide experiment results that demonstrate the proposed Attentive TPP capture periodic patterns better than the baselines. To validate that the datasets used for Table 2 have strong periodic patterns, we provide visualization in Figure 4. Sinusoidal wave dataset has a periodicity for every 4π as shown in Figure 4a, Uber datset has weekly periodic pattern shown in Figure 4b, and NYC Taxi dataset has daily pattern as shown in Figure 4c.

[2]https://www.kaggle.com/datasets/fivethirtyeight/uber-pickups-in-new-york-city/metadata
[3]http://www.andresmh.com/nyctaxitrips/

I Hyperparameters

We use the feature dimension of 96, 72, and 64 for the intensity-free (Shchur et al., 2020), neural flow (Biloš et al., 2021), and THP$^+$, respectively, as the numbers of parameters fall in the range of 50K and 60K with those dimensions.

For the Meta TPP, we use 64 for the dimension of the latent variable z, and 32 samples to approximate the ELBO for variantional inference. As the variance of variational inference is generally low, 32 samples are enough to have stable results. In inference, we increase the sample size to 256 to have more accurate approximation. For the Attentive TPP, we use 1-layer self-attention for the cross-attention path, and fix the local history window size to 20.

For training, we use a batch size of 16 throughout all the models, and optimize with an Adam optimizer for a grid search for learning rate and weight decay described in Section 5.