# TEST-TIME ADAPTATION FOR EVENT PREDICTION VIA LIGHTWEIGHT ADAPTERS

**Shivam Grover** *
Queen's University, Canada
shivam.grover@queensu.ca

**Hossein Hajimirsadeghi, Zhitian Zhang, Edward J. Smith & Alexander Pashevich**
RBC Borealis, Canada
{hossein.hajimirsadeghi, andy.zhang}@rbc.com
{edward.smith, alexander.pashevich}@rbc.com

## ABSTRACT

Event prediction is central to applications from e-commerce to finance, yet real-world event streams are highly non-stationary, making deployment of frozen pre-trained models brittle. We propose **E**vent prediction **T**est-**T**ime **A**daptation (ETTA), a model-agnostic lightweight framework that enables event-prediction models to adapt on-the-fly during inference, without retraining the model itself. Unlike model fine-tuning at test-time, ETTA introduces compact temporal and logit calibration modules that calibrate the inter-event timestamps and event type logits respectively. These adapters can be seamlessly applied to both neural temporal point process (TPP) models and large language models (LLMs). On five real-world benchmarks, ETTA consistently improves TPP models, achieving up to 20.3% reduction in RMSE and 4.9% higher accuracy, while for LLM-based event prediction it yields up to 23.1% RMSE reduction and 9.0% accuracy improvement. Together, these results establish test-time adaptation, and ETTA in particular, as a powerful, general framework for robust event prediction under distribution shift.

## 1 INTRODUCTION

Event prediction is a foundational problem underpinning critical applications in domains ranging from finance and healthcare to e-commerce and social media (Zhou et al., 2025). A unifying challenge in these applications is that event streams are both irregular and non-stationary: events occur at uneven intervals, and the distribution of event types evolves over time. Traditional event prediction models do not explicitly handle these shifts after pretraining, and their performance often degrades once the test data's distribution shifts (Zhou et al., 2025). This limitation persists even for advanced neural temporal point process (TPP) models, which capture complex dependencies but lack mechanisms to adapt when faced with real-world distribution drift.

Our key observation is that ground truth in event sequences is revealed sequentially at test time. Each sequence therefore provides its own adaptation set in the form of the observed prefix, offering an opportunity to recalibrate the model before predicting the next event. Rather than relying on brittle frozen predictors, one can exploit these per-sequence signals to perform lightweight, on-the-fly adaptation. In addition to improving prediction quality, we show that adapter-based calibration is consistently faster than test-time fine-tuning in per-sequence wall-clock runtime (Table 3), supporting its practicality in latency-sensitive settings. This perspective opens a new paradigm for robust event prediction, enabling models to maintain accuracy and stability in the presence of temporal non-stationarity.

In this paper we present ETTA (**E**vent prediction **T**est-**T**ime **A**daptation), a model-agnostic framework for efficiently adapting pretrained event-prediction models at test-time. Specifically, ETTA keeps the pretrained model frozen at all times, and attaches two learnable lightweight adapters: (1)

*Work done during an internship at RBC Borealis.

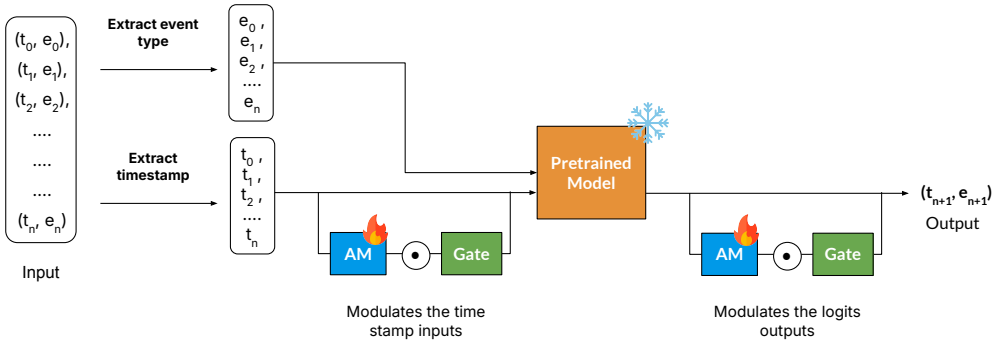


Figure 1: Overview of our **E**vent prediction **T**est-**T**ime **A**daptation (ETTA). A temporal adapter (left) calibrates the event timestamp before they enter the frozen pretrained model, while an event adapter (right) adjusts the output logits after the model prediction. During inference, the observed prefix of a sequence is used to optimize these adapters, enabling the model to adapt dynamically to distribution shifts and produce robust predictions for the next event timestamp and type.

a temporal adapter before the pretrained model to calibrate the timestamp information, and (2) an event adapter after the pretrained model to calibrate output logits. At test-time, the observed prefix of each event sequence is used to optimize these adapters, capturing the dynamic distribution shifts the sequence undergoes over time and helping the model be aligned to the current sequence's distribution. This design exploits the sequential revelation of ground truth while preserving the stability of the pretrained model, enabling robust performance under temporal distribution shifts.

To validate our approach, we conduct extensive experiments across a diverse set of TPP architectures and benchmark datasets. Specifically, we evaluate ETTA on four representative TPP models including NHP (Mei & Eisner, 2017), RMTPP (Du et al., 2016), SAHP (Zhang et al., 2020), and AttNHP (Yang et al., 2022). We use the five standard datasets provided in the EasyTPP benchmark (Xue et al., 2023a). Across these combinations, ETTA yields consistent gains, with 20.3% lower RMSE and 4.9% higher accuracy compared to the baseline.

We further extend ETTA to LLMs for event prediction, building on recent work showing that LLMs can effectively model event sequences through textual prompts (Shi et al., 2023b). In these experiments, we operate in a zero-shot setting, meaning that the LLM is not trained directly on the given event prediction dataset beforehand. Instead, we directly apply ETTA at inference, calibrating adapters on each test sequence, while keeping the LLM frozen and using the system prompt to specify the event prediction task. On the same benchmark datasets, ETTA achieves up to 23.1% reductions in RMSE and 9.0% accuracy gains demonstrating that our framework generalizes beyond specialized TPP models to powerful pretrained LLMs.

While adapter-based test-time adaptation has been explored for conventional time-series forecasting, to our knowledge ETTA is the first framework to bring test-time adaptation to continuous-time event prediction, which requires jointly forecasting irregular timestamps and discrete event types; this setting makes a direct reuse of time-series-forecasting test-time-adaptation (TSF-TTA) methods infeasible and motivates our structural decoupling of temporal calibration and event/logit calibration. Moreover, ETTA provides the first demonstration of *zero-shot* LLM test-time adaptation for event prediction, enabling sequence-specific temporal drift correction without any backbone fine-tuning.

Our contributions can be summarized as follows:

- We propose ETTA, a lightweight and model-agnostic framework that adapts pre-trained event-prediction models at test-time using two adapters for calibrating the input timestamps and output logits respectively.
- Rigourous experiments on multiple TPP benchmarks show that applying ETTA at test time yields substantial performance gains, while keeping the pretrained model frozen and requiring only lightweight adapter training.
- We further demonstrate that ETTA generalizes beyond specialized TPPs by extending it to pretrained large language models (LLMs) in a zero-shot setting, where it leverages

their broad semantic knowledge while adapting at test time to sequence-specific dynamics, achieving significant accuracy gains and RMSE reductions.

# 2 METHODOLOGY

In this section, we introduce our proposed approach for TTA in event prediction. We begin by formalizing the event prediction task and describing how continuous-time event sequences are typically modeled. We then present test-time fine-tuning as a straightforward baseline and discuss its limitations. Building on these insights, we develop our **E**vent prediction **T**est-**T**ime **A**daptation (ETTA) framework, which augments a frozen pretrained backbone with lightweight temporal and event adapters. Finally, we describe how ETTA is instantiated for both TPP models and LLMs. To learn more about related works, see Appendix A.

## 2.1 BACKGROUND

**Event Prediction.** Given a continuous-time event sequence $\mathcal{S}_{1:N} = \{(t_1, e_1), (t_2, e_2), \ldots, (t_N, e_N)\}$, where each event is represented by its occurrence time $t_i \in \mathbb{R}^+$ and event type $\epsilon_i \in \mathcal{E}$, the prediction task has two aspects: (1) predicting ***when***, i.e. the timestamp $t_{N+1}$ of the next event, and (2) predicting ***what***, i.e. the type $e_{N+1}$ of the next event. Following Xue et al. (2023a), we represent temporal dynamics via inter-event timestamps $\Delta t_i = t_i - t_{i-1}$ with the convention $t_0 = 0$. This formulation allows the model to focus on relative timing rather than absolute timestamps, making the representation more robust to shifts. In this setting, given a history of events $H_{N-1} = \{(\Delta t_1, e_1), \ldots, (\Delta t_{N-1}, e_{N-1})\}$, the model learns conditional distributions over the next event types and timestamps. The temporal head models the distribution of the next inter-event time, while the categorical head models a probability distribution over event types. The model parameters $\theta$ are trained by maximizing the likelihood of observed sequences, or equivalently minimizing the negative log-likelihood across both time and event predictions. Model predictions are evaluated against the ground truth $(\Delta t_N, e_N)$ using a regression metric such as RMSE for the timestamp and accuracy for the event type.

At test time, the model observes the prefix $\mathcal{S}_{1:N-1}$ and is asked to predict the final event $(t_N, e_N)$. Using inter-event timestamp representation, the model observes $\{(\Delta t_1, e_1), \ldots, (\Delta t_{N-1}, e_{N-1})\}$ as input and need to forecast $(\Delta t_N, e_N)$. From this single sequence, we can construct multiple supervised pairs for adaptation by exploiting the ground truth that is presented to the model sequentially. Concretely, for each $j < N$, we define a history prefix $H_j = \{(\Delta t_1, e_1), \ldots, (\Delta t_j, e_j)\}$, with corresponding next-step target $y_j = (\Delta t_{j+1}, e_{j+1})$. Collecting these pairs, we define a set of subsequences $\mathcal{D}_{\text{test}}(\mathcal{S}) = \big\{(H_j, y_j) : j = 1, \ldots, N-2\big\}$, which acts as an in-test training set. Using the observed subsequence-target pairs in this set, we can adapt the model before making the final prediction of $y_{N-1} = (\Delta t_N, e_N)$.

**Test-time Fine Tuning.** A straightforward baseline for leveraging the subsequences defined above is to fine-tune the pretrained model directly at test time. Given a test sequence $\mathcal{S}_{1:N}$ of length $N$, we construct the adaptation set $\mathcal{D}_{\text{test}}(\mathcal{S})$ consisting of subsequence–target pairs $(H_j, y_j)$ for $j < N$. The pretrained model is then fine-tuned on $\mathcal{D}_{\text{test}}(\mathcal{S})$ , updating all parameters until the last subsequence has been processed. After this fine-tuning stage, the adapted model is used to predict the final event $(\Delta t_N, e_N)$. In practice, each dataset contains many independent test sequences, and adaptation is performed separately on each of them. To avoid leakage of future data from one test sequence to the other, the model parameters are reset to their original pretrained values before processing each new sequence. In this setup, the amount of adaptation depends solely on the number of subsequences available within the given test sequence, and no information is shared across sequences. This method leverages the full expressivity of the model to adjust to local distributional shifts. However, it is computationally expensive, since every forward pass during adaptation must be followed by backpropagation through the entire model; and updating the model's weights at each step runs the risk of catastrophic forgetting.

## 2.2 EVENT PREDICTION TEST-TIME ADAPTATION (ETTA)

To enable efficient adaptation under distribution shift, we propose **E**vent prediction **T**est-**T**ime **A**daptation (ETTA), which keeps the pretrained backbone frozen and augments it with lightweight adapters for targeted calibration. ETTA employs two complementary modules: (1) a *temporal*

*adapter* $A_\tau$, which corrects for shifts in the temporal characteristics of the sequence, and (2) an *event class adapter* $A_e$, which adjusts the event-type representation or predictions to account for changes in the event distribution. Both adapters are designed to be small, efficient, and easily optimized during test time, allowing the model to adapt to distributional shifts without updating the entire backbone. To regulate their influence, we employ gating mechanisms that interpolate between the original representation and the adapter outputs. A general formulation of the adapter being applied to a vector is

$$\tilde{x} = (1 - \alpha_x)\, x + \alpha_x \cdot A_x(x; \phi), \tag{1}$$

where $x$ can be either the the inter-event time data or the event classification logits, $A_x(\cdot; \phi)$ is its corresponding adapter with trainable parameters $\phi$, and $\alpha_x \in [0, 1]$ is a learned gate. This residual-style gating yields stable, controlled reprogramming: since the backbone is frozen, the adapters act as lightweight corrective layers, and the gate regulates update strength to reduce overfitting on test-time subsequences. Our ablations in Appendix B.2 (Tables 4 and 5) further show that the temporal and logit adapters address different forms of drift and combining both adapters yields the most consistent gains. Below, we describe how ETTA is applied to TPP models and LLMs.

**ETTA for TPP models.** In the TPP setting, each sequence is represented by inter-event timestamps and event types. At test time, we first extract the inter-event timestamps $\Delta t_i = t_i - t_{i-1}$ from the observed sequence prefix $\mathcal{S}_{1:N-1}$. These temporal features are passed through the temporal adapter, which outputs calibrated values via a gated update:

$$\Delta\tilde{t}_i = (1 - \alpha_\tau)\, \Delta t_i + \alpha_\tau \cdot A_\tau(\Delta t_i; \phi_\tau) \tag{2}$$

where $A_\tau(\cdot; \phi_\tau)$ denotes the temporal adapter with learnable parameters $\phi_\tau$, and $\alpha_\tau \in [0, 1]$ is its learnable gating parameter that controls the strength of the calibration. The calibrated timestamps are then merged back with the corresponding event types to form the adapted input sequence $\tilde{H}_{N-1} = \{(\Delta\tilde{t}_1, e_1), \ldots, (\Delta\tilde{t}_{N-1}, e_{N-1})\}$. This calibrated sequence is processed by the pretrained model backbone, which remains frozen during test time, and predicts both the next inter-event time $\Delta\hat{t}_N$ and raw event logits $\ell_N$ as $(\Delta\hat{t}_N, \ell_N) = f_\theta(\tilde{H}_{N-1})$, where $f_\theta$ represents the pretrained model. To address potential shifts in the event distribution, the event class adapter operates on the logits. Its effect is also gated to control the influence of the original backbone prediction and the calibrated output:

$$\tilde{\ell}_N = (1 - \alpha_e)\, \ell_N + \alpha_e \cdot A_e(\ell_N; \phi_e) \tag{3}$$

where $A_e(\cdot; \phi_e)$ is the event type adapter with learnable parameters $\phi_e$, and $\alpha_e \in [0, 1]$ is its learnable gating parameter. The final prediction for the next event type is then obtained as $p(e_N \mid \tilde{H}_{N-1}) = \mathrm{softmax}(\tilde{\ell}_N)$.

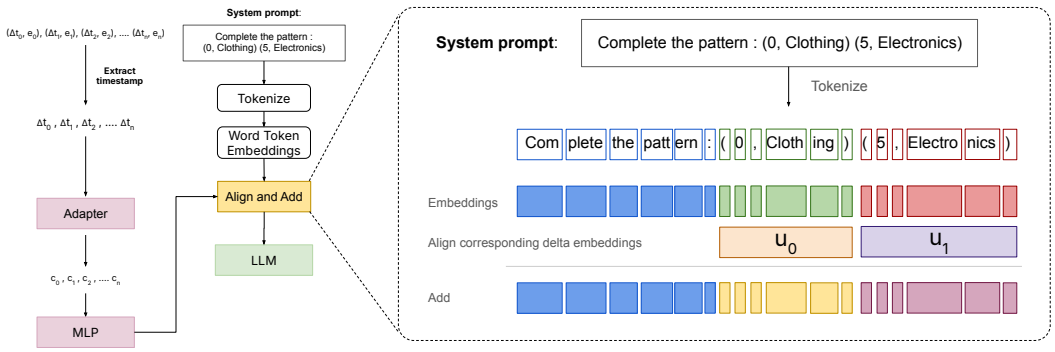


Figure 2: Overview of applying ETTA to LLMs for event prediction. Inter-event times are extracted from the input sequence, calibrated with a temporal adapter, and mapped to the LLM embedding space via an MLP. These calibrated vectors are aligned with the corresponding token embeddings of the structured prompt and added to them, producing modified embeddings that are fed into the frozen LLM backbone. The adapted representation, combined with the system prompt and allowed event types, enables the LLM to generate predictions that are further refined by the event class adapter following the LLM (not shown on the figure).

**ETTA for LLMs.** The high-level overview of the proposed framework is shown in Figure 2. Here, we begin by outlining the general methodology for performing event prediction with an LLM

before describing how ETTA is applied in this setting. Given an observed sequence $H_{1:N-1} = \{(\Delta t_1, e_1), \ldots, (\Delta t_{N-1}, e_{N-1})\}$ and an exhaustive list of allowed event types $\mathcal{E} = \{\epsilon_1, \ldots, \epsilon_M\}$, we construct a structured prompt $\pi(S_{N-1}, \mathcal{E})$ consisting of (i) the **system prompt** that describes the event sequence and the dataset, and provides instructions on what the task is and what the output should look like, (ii) **allowed event types** which enumerates the allowed event types $\mathcal{E}$, and (iii) the **input sequence** which includes the input time series as a list of tuples $(\Delta t_i, e_i)$. A sample prompt to the LLM is given below for the Amazon dataset.

**System Prompt:** You are given a sequence of past events from the Amazon dataset. Each event is represented as a tuple of the form (inter_event_time, event_type), where inter_event_time is the inter-event time since the previous event, and event_type is the category of the event. Your task is to predict the next event in the sequence. The output must be in the exact format: "(inter_event_time, [event_type])".

**Allowed Event Types:** "Clothing", "Shoes", "Accessories", "Jewelry", "Novelty & More", "Watches", "Uniforms", "Work & Safety", "Luggage & Travel Gear", "Costumes & Accessories", "Girls", "Boys", "Shoe", "Jewelry & Watch Accessories", "Baby", "Handbags & Wallets", "Surf", "Skate & Street", "Contemporary & Designer".

**Input Sequence:** (2.1, Clothing) (0.5, Shoes) (3.7, Accessories) (1.2, Jewelry)

The LLM then generates a prediction in the required format $(\Delta\hat{t}_N, \hat{e}_N)$, where $\Delta\hat{t}_N$ is parsed as a numeric inter-event time and $\hat{e}_N$ is constrained to one of the allowed event types. The output is processed through simple string parsing to extract both the predicted time and event label, which are subsequently used for evaluation against the ground truth.

However, since we do not train the LLM explicitly on event prediction tasks, or on our datasets, the LLM lacks the necessary temporal understanding and capabilities to capture the dynamic non-stationarity in continuous-time event sequences. Unlike TPP models that operate directly on numerical inter-event times, LLMs process tokenized text, so the generic adapter formulation cannot be applied as-is. Consequently, we apply the necessary changes to ETTA to make it compatible with a generic LLM architecture.

For ***timestamp adaptation***, we begin by extracting the sequence of inter-event times $\{\Delta t_1, \ldots, \Delta t_{N-1}\}$ from the observed prefix $\mathcal{S}_{1:N-1}$. As shown in Figure 2, each $\Delta t_i$ is passed through the temporal adapter $A_\tau$, which outputs a calibration offset $c_i = A_\tau(\Delta t_i; \phi_\tau)$. This offset is then mapped through a small multi-layer perceptron (MLP) to the LLM embedding dimension: $u_i = \mathrm{MLP}(c_i) \in \mathbb{R}^d$, where $d$ is the embedding size. In parallel, the structured prompt $\pi(\mathcal{S}_{N-1}, \mathcal{E})$ is tokenized, and its word-token embeddings (wte) are obtained from the LLM as $z_{1:L} = \mathrm{wte}(\pi)$. For each event tuple $(\Delta t_i, e_i)$, we add the predicted $u_i$ to the token span $z_i$ corresponding to the tuple $(\Delta t_i, e_i)$ and update it as $\tilde{z}_i = z_i + u_i$. Structural tokens (e.g., brackets, commas, whitespace) and system prompt tokens are excluded from calibration, ensuring that adaptation is applied only to semantic tokens representing timestamps and event types. The resulting modified embedding sequence $\tilde{z}_{1:L}$ is then passed through the remaining layers of the transformer backbone to produce logits.

For ***event class adaptation*** with an LLM, the role is not to rewrite tokens directly, but to re-weight the event logits so that the correct tokens are favored while the others are not. We adapt only the logits relevant to the event class, and this is aided by the output format enforced by the system prompt shown earlier. More specifically, in the system prompt, we ask the LLM to output in the format "(inter event time, [event type])". This design ensures that the tokens for '**[**' and '**]**' explicitly delimit the start and end of the event type, making it straightforward to extract the relevant logits for calibration. At test time, the LLM produces logits $\ell$ over its entire vocabulary $V$. We restrict these to the allowed event types $\mathcal{E} = \{\epsilon_1, \ldots, \epsilon_M\}$, where each $\epsilon_m$ denotes a candidate event type from the vocabulary. For each possible event $\epsilon_m$, we compute its event score $s_m$ as follows: if $\epsilon_m$ corresponds to a single token, $s_m$ equals to the associated logit; if $\epsilon_m$ is a multi-token phrase, $s_m$ is obtained as the aggregated log-probability of its constituent tokens. The event class adapter $A_e$ then calibrates these event-level scores through a gated update:

$$\tilde{s}_m = (1 - \alpha_e)\, s_m + \alpha_e \cdot A_e(s_m; \phi_e) \tag{4}$$

where $\alpha_e \in [0, 1]$ is a learnable gate and $\phi_e$ are the adapter parameters. The calibrated scores $\{\tilde{s}_m\}$ are normalized across $\mathcal{E}$ to yield the final event prediction distribution:

$$p(e_m \mid H_k) = \frac{\exp(\tilde{s}_m)}{\sum_{j=1}^{M} \exp(\tilde{s}_j)} \tag{5}$$

Thus, ETTA acts by shifting probability mass between competing event classes, bringing down incorrect options and elevating the correct ones. During test-time adaptation, we backpropagate through the (frozen) LLM computation graph, but only the temporal and event adapters are updated via these differentiable embedding- and score-level operations.

## 3 EXPERIMENTS

In this section, we evaluate ETTA on five benchmarks with both TPP and LLM backbones. We first describe datasets, backbones, baselines, metrics, and implementation details, then report the main performance results. Additionally, we include runtime comparisons (Table 3) and adapter ablations (Tables 4 and 5 in Appendix B.2) to assess the efficiency and contribution of each component.

**Datasets.** We conduct our experiments on five diverse datasets for event prediction from the popular EasyTPP benchmark (Xue et al., 2023a):

- **Amazon** (Ni et al., 2019) which contains product review events, with the task to predict the timestamp and category of the next reviewed product. Event categories correspond to 16 product categories (e.g., Books, Electronics, Clothing). Labels are available as discrete product-type indices.
- **Stackoverflow** (Jure, 2014) with records of assignment of badges to users over time. Each badge assignment is treated as an event, and the task involves predicting the next badge type and its timestamp. The dataset includes 22 categories corresponding to different badge types, and only badge indices are available as labels.
- **Taobao** (Xue et al., 2022) which contains user click events from the Taobao e-commerce platform. Each event represents a user clicking on an item, and the task involves predicting the timestamp and category of the clicked item. The dataset provides 20 item categories, labelled as indices without explicit semantic names.
- **Taxi** (Whong, 2014) which contains GPS-based records of New York City taxi pickups and dropoffs. Each event type is defined by a (borough, trip-start/trip-end) pair (10 event types total), and each sequence corresponds to a single driver. We represent timestamps as inter-event times $\Delta t$; for Taxi, $\Delta t$ for a trip-start event is the taxi's idle time since the preceding trip-end, while $\Delta t$ for a trip-end event is the trip duration.
- **Retweet** (Zhou et al., 2013) which contains social network cascade events where a user retweets a post. Each event involves predicting the timestamp and which user (or user type) retweets next. The dataset simplifies users into 3 categorical classes.

**Backbones.** To benchmark ETTA with strong TPP baselines, we adopt four backbones: RNN based neural Hawkes process (NHP) (Mei & Eisner, 2017), recurrent marked temporal point process (RMTPP) (Du et al., 2016), as well as two attention-based models including self-attentive Hawkes pocess (SAHP) (Zhang et al., 2020) and attentive neural Hawkes process (AttNHP) (Yang et al., 2022).

To further evaluate ETTA with LLMs for event prediction, we experiment with four representative LLM backbones that vary in scale and architectural design. These include Qwen 3 (0.6B) (Zhang et al., 2025), a compact model suitable for efficiency-focused experiments; Llama 2 (7B) Touvron et al. (2023), a widely used open-source foundation model with strong generalization; Llama 3 (8B) (Grattafiori et al., 2024), the version of LLama with improved instruction following and sequence reasoning; and Mistral (7B) (Jiang et al., 2023), a dense transformer architecture optimized for competitive downstream performance. This selection spans lightweight to mid-sized LLMs, allowing us to assess the impact of model capacity on asynchronous event modeling, while also reflecting our computation resource constraints that prevent us from using much larger models.

We compare our proposed method against two key baselines for each backbone: the frozen pretrained model (Base) and **T**est-**T**ime **F**ine-**T**uning (TTFT) described in Section 2.1.

Table 1: Performance comparison of event prediction using TPP models with a pretrained frozen model, with TTFT and with ETTA (our approach). For each backbone, we show the results without any adaptation (Base), and with TTFT and ETTA adaptations. A higher accuracy, and a lower RMSE are better. **Best** results for each dataset are highlighted with bold. Percentage of improvement (Imp%) denotes the relative improvement of ETTA compared to Base. All experiments were ran for three different random seeds and we report their mean.

| Model | Metric | Amazon | | | | Retweet | | | | StackOverflow | | | | Taobao | | | | Taxi | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Base | TTFT | ETTA | Imp% | Base | TTFT | ETTA | Imp% | Base | TTFT | ETTA | Imp% | Base | TTFT | ETTA | Imp% | Base | TTFT | ETTA | Imp% |
| NHP | Accuracy ↑ | 0.3332 | 0.3378 | 0.3495 | 4.88% | 0.6000 | 0.5986 | 0.6109 | 1.82% | 0.4500 | 0.4589 | **0.4619** | 2.64% | 0.4580 | 0.4612 | 0.4678 | 2.14% | 0.9150 | 0.9012 | 0.9089 | -0.67% |
| | RMSE ↓ | 0.6280 | 0.6190 | 0.6070 | 3.34% | 22.3128 | 21.7364 | 20.0320 | 10.22% | 0.1379 | 0.1398 | 0.1312 | 4.86% | 0.5321 | 0.5319 | 0.5197 | 2.33% | 0.3895 | 0.3623 | 0.3103 | 20.34% |
| SAHP | Accuracy ↑ | 0.3304 | 0.3367 | 0.3473 | 5.13% | 0.5840 | 0.5912 | 0.5987 | 2.52% | 0.4390 | 0.4389 | 0.4533 | 3.26% | 0.4540 | 0.4522 | 0.4681 | 3.11% | 0.9025 | 0.9078 | **0.9189** | 1.82% |
| | RMSE ↓ | 0.6230 | 0.6270 | 0.6110 | 1.93% | 24.0123 | 22.4932 | 20.4219 | 14.95% | 0.1378 | 0.1354 | **0.1294** | 6.07% | 0.5300 | 0.5215 | 0.5184 | 2.19% | 0.3506 | 0.3314 | **0.3098** | 11.63% |
| RMTPP | Accuracy ↑ | 0.3258 | 0.3287 | 0.3345 | 2.68% | 0.5590 | 0.5619 | 0.5688 | 1.75% | 0.4270 | 0.4289 | 0.4388 | 2.76% | 0.4420 | 0.4489 | 0.4612 | 4.34% | 0.9049 | 0.9123 | 0.9127 | 0.86% |
| | RMSE ↓ | 0.6215 | 0.6195 | 0.6105 | 1.77% | 22.3600 | 22.4364 | 21.9520 | 1.82% | 0.1375 | 0.1362 | 0.1333 | 3.05% | 0.5389 | 0.5319 | **0.5023** | 6.79% | 0.3790 | 0.3670 | 0.3510 | 7.39% |
| AttNHP | Accuracy ↑ | 0.3511 | 0.3507 | **0.3599** | 2.50% | 0.5990 | 0.5998 | **0.6201** | 2.17% | 0.4480 | 0.4495 | 0.4612 | 2.95% | 0.4630 | 0.4698 | **0.4719** | 1.92% | 0.9129 | 0.9141 | 0.9187 | 0.64% |
| | RMSE ↓ | 0.6230 | 0.6210 | **0.6049** | 2.91% | 22.1400 | 21.7314 | **19.1320** | 13.59% | 0.1377 | 0.1371 | 0.1309 | 4.94% | 0.5312 | 0.5291 | 0.5198 | 2.15% | 0.3780 | 0.3470 | 0.3270 | 13.49% |

Table 2: Performance comparison of event prediction using LLMs with and without ETTA (our approach). For each backbone, we show the results without any adaptation (Base), and with TTFT and ETTA adaptations. A higher accuracy, and a lower RMSE are better. **Best** results for each dataset are highlighted with bold. Percentage of improvement (Imp%) denotes the relative improvement of ETTA compared to Base. All experiments were ran for three different random seeds and we report their mean.

| Model | Metric | Amazon | | | Retweet | | | StackOverflow | | | Taobao | | | Taxi | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Base | ETTA | Imp% | Base | ETTA | Imp% | Base | ETTA | Imp% | Base | ETTA | Imp% | Base | ETTA | Imp% |
| Qwen 3 0.6B | Accuracy ↑ | 0.2631 | 0.2738 | 4.07% | 0.5233 | 0.5495 | 5.01% | 0.3671 | 0.3918 | 1.36% | 0.3989 | 0.3993 | 0.10% | 0.7510 | 0.8121 | 8.14% |
| | RMSE ↓ | 0.6981 | 0.6572 | 5.86% | 39.1823 | 30.1419 | 23.07% | 0.1455 | 0.1398 | 3.92% | 0.5876 | 0.5592 | 4.83% | 0.3988 | 0.3586 | 10.08% |
| Llama 2 7B | Accuracy ↑ | 0.2981 | 0.3101 | 4.03% | 0.5389 | 0.5301 | -1.63% | 0.3905 | 0.4129 | 5.74% | 0.3991 | **0.4231** | 6.01% | 0.8022 | 0.8195 | 2.16% |
| | RMSE ↓ | 0.6544 | 0.6399 | 2.22% | 28.9819 | 24.1298 | 16.74% | 0.1481 | 0.1492 | -0.74% | 0.5698 | 0.5512 | 3.26% | 0.3743 | 0.3614 | 3.45% |
| Llama 3 8B | Accuracy ↑ | 0.3128 | **0.3235** | 3.42% | 0.5482 | **0.5729** | 4.51% | 0.4645 | **0.4698** | 1.14% | 0.4198 | 0.4217 | 0.45% | 0.8819 | **0.8950** | 1.49% |
| | RMSE ↓ | 0.6819 | **0.6091** | 10.68% | **21.9878** | 22.3198 | -1.51% | 0.1440 | 0.1398 | 2.92% | 0.5501 | **0.5488** | 0.24% | 0.3697 | **0.3549** | 4.00% |
| Mistral 7B | Accuracy ↑ | 0.2744 | 0.2989 | 8.93% | 0.5197 | 0.5313 | 2.23% | 0.3877 | 0.4012 | 3.48% | 0.3812 | 0.4155 | 9.00% | 0.7932 | 0.8188 | 3.23% |
| | RMSE ↓ | 0.7124 | 0.6781 | 4.81% | 25.1901 | 23.1927 | 7.93% | 0.1418 | **0.1356** | 4.37% | 0.6194 | 0.6011 | 2.95% | 0.3890 | 0.3711 | 4.60% |

**Evaluation Metrics.** Event prediction involves two tasks: predicting the next event type and its inter-event time. Accordingly, we report accuracy for event type prediction and root mean squared error (RMSE) for inter-event time.

**Implementation Details.** For pretraining the TPP backbones as well as during TTFT, we use the implementations provided by Xue et al. (2023a). We keep data splits and model hyperparameters as prescribed by the authors of EasyTPP benchmark. For ETTA, the only additional hyperparameter is the adaptation learning rate, which we tune by searching over {1e-3, 1e-4, 1e-5, 5e-6} and selecting the value that performs best on the validation set.

All of our LLM-based experiments are performed in a zero-shot setting, i.e. we do not employ the train set and only use the test set. We use the transformer library from huggingface [1] for the implementation.

**Main performance results.** Table 1 reports results of applying ETTA to four TPP backbones across five benchmark datasets. ETTA delivers consistent gains across all settings, outperforming both the frozen pretrained model and the stronger TTFT baseline. Relative to the frozen model, ETTA achieves up to 4.9% accuracy improvement on Amazon and as much as 20.3% reduction in RMSE on Taxi. Moreover, ETTA consistently surpasses TTFT across 39 out of 40 metrics, despite TTFT updating all model parameters. We attribute this advantage to the gating mechanism in ETTA, which enables the adapters to modulate their influence based on the dynamics of each input sequence. The benefits are most pronounced on datasets with highly irregular temporal patterns such as Taxi and Retweet, where distribution shifts are strongest, while improvements are more modest on relatively more regular datasets like Amazon. These results demonstrate that lightweight, sequence-specific adaptation with ETTA is both more robust and more efficient than full model fine-tuning.

Furthermore, to qualitatively illustrate the model's behavior, we provide autoregressive sequence predictions for two representative examples from the Amazon dataset in Appendix B.1.

[1] https://huggingface.co/docs/transformers/

Table 2 reports results of applying ETTA to LLM backbones across the five benchmark datasets. ETTA consistently improves over the frozen pretrained LLMs, achieving up to 9.0% accuracy gains on Taobao and as much as 23.1% RMSE reduction on Retweet. These gains are obtained without updating any backbone parameters, underscoring the effectiveness of lightweight adapter-based calibration even in large-scale models. While overall LLM performance remains below that of specialized TPP models—likely due to the strict zero-shot setting where LLMs are not trained on these datasets, ETTA substantially narrows the gap. In several cases, such as StackOverflow and Taxi, adapted LLMs approach or match the performance of fully trained TPPs, highlighting the potential of TTA to unlock competitive event prediction from general-purpose language models even in the zero-shot setting.

The key insight from these experiments is the universality of ETTA. Unlike approaches tied to a specific architecture, it can be applied to both specialized temporal models and general-purpose LLMs. In all cases, it delivers measurable gains, underscoring its role as a model-agnostic solution for event prediction.

**Runtime Comparisons.** To substantiate the claim that ETTA is compute-efficient, Table 3 reports the average wall-clock time (seconds per test sequence) required to adapt and make a prediction for a single test sequence using ETTA versus TTFT, across four TPP backbones and two datasets; we also report the "No adapt" inference time (i.e., a single forward pass of the frozen backbone) for reference. As expected, both TTFT and ETTA introduce overhead compared to the frozen model since they perform per-sequence optimization at inference time; however, ETTA achieves consistent reductions in per-sequence adaptation time compared to TTFT on both StackOverflow and Taxi, while simultaneously improving accuracy and RMSE (Tables 1 and 2). These results confirm that ETTA is not only parameter-efficient, but also compute-efficient relative to full fine-tuning, making it a practical choice for real-time or latency-sensitive deployment scenarios.

Table 3: Runtime comparison of per-sequence test-time adaptation on **StackOverflow** and **Taxi**. We report average wall-clock seconds per test sequence (adaptation + prediction) for no adaptation (No adapt), Test-Time Fine-Tuning (TTFT), and ETTA. Lower is better; best (fastest) among TTFT and ETTA is bolded.

| Model | StackOverflow | | | Taxi | | |
|---|---|---|---|---|---|---|
| | No adapt | TTFT | ETTA | No adapt | TTFT | ETTA |
| AttNHP | 0.01285 | 2.39470 | **1.65548** | 0.00975 | 1.25730 | **1.09610** |
| RMTPP | 0.01095 | 0.47781 | **0.44390** | 0.01225 | 0.35230 | **0.22421** |
| SAHP | 0.01800 | 0.94454 | **0.80819** | 0.01050 | 0.32573 | **0.17888** |
| NHP | 0.03975 | 0.76404 | **0.66755** | 0.01125 | 0.72281 | **0.65678** |

**Adapter Ablations.** To better understand the role of each component in ETTA, we ablate the temporal adapter and the logits adapter, evaluating (i) temporal-only, (ii) logits-only, and (iii) the full dual-adapter ETTA. Tables 4 and 5 in Appendix B.2 show that the two adapters play complementary roles: the temporal adapter primarily improves timestamp prediction (lower RMSE), while the logits adapter primarily improves event-type prediction (higher accuracy).

## 4 CONCLUSION

We presented a model-agnostic and efficient framework ETTA for event-prediction TTA, which learns two lightweight adapters during test-time: a temporal adapter to calibrate inter-event times and an event adapter to calibrate output logits. Crucially, ETTA exploits the observed prefix of each test sequence as its own adaptation set, enabling targeted, sequence-specific calibration without retraining the full model. Additionally, we extend the ETTA framework to work with LLM-based event prediction in a zero-shot setting that leverages the broad semantic knowledge of pretrained LLMs, while adapting at test time to sequence-specific dynamics. Through rigorous experiments on a diverse set of datasets, we show that ETTA consistently improves the predictions across all TPP and LLM-based backbones, yielding up to 20.3% reductions in RMSE and 5.1% accuracy improvements for TPP backbones, and up to 23.1% reductions in RMSE and 9.0% accuracy improvements.

## REFERENCES

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pp. 4171–4186, 2019.

Nan Du, Hanjun Dai, Rakshit Trivedi, Utkarsh Upadhyay, Manuel Gomez-Rodriguez, and Le Song. Recurrent marked temporal point processes: Embedding event history to vector. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1555–1564, 2016.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Shivam Grover and Ali Etemad. Shift-aware test time adaptation and benchmarking for time-series forecasting. In *Second Workshop on Test-Time Adaptation: Putting Updates to the Test! at ICML 2025*, 2025. URL `httpsc://openreview.net/forum?id=a399SmgWGl`.

Shubham Gupta, Thibaut Durand, Graham Taylor, et al. Last stop for modeling asynchronous time series. *arXiv preprint arXiv:2502.01922*, 2025.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b. *CoRR*, abs/2310.06825, 2023. doi: 10.48550/ARXIV.2310.06825. URL `https://doi.org/10.48550/arXiv.2310.06825`.

Leskovec Jure. Snap datasets: Stanford large network dataset collection. *Retrieved December 2021 from http://snap. stanford. edu/data*, 2014.

HyunGi Kim, Siwon Kim, Jisoo Mok, and Sungroh Yoon. Battling the non-stationarity in time series forecasting via test-time adaptation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 17868–17876, 2025.

Quyu Kong, Yixuan Zhang, Yang Liu, Panrong Tong, Enqi Liu, and Feng Zhou. Language-tpp: Integrating temporal point processes with language models for event analysis. *arXiv preprint arXiv:2502.07139*, 2025.

Zefang Liu and Yinzhu Quan. Tpp-llm: Modeling temporal point processes by efficiently fine-tuning large language models. *arXiv preprint arXiv:2410.02062*, 2024.

Heitor R Medeiros, Hossein Sharifi-Noghabi, Gabriel L Oliveira, and Saghar Irandoust. Accurate parameter-efficient test-time adaptation for time series forecasting. *arXiv preprint arXiv:2506.23424*, 2025.

Hongyuan Mei and Jason M Eisner. The neural hawkes process: A neurally self-modulating multivariate point process. *Advances in neural information processing systems*, 30, 2017.

Jianmo Ni, Jiacheng Li, and Julian McAuley. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pp. 188–197, 2019.

Takahiro Omi, Kazuyuki Aihara, et al. Fully neural network based model for general temporal point processes. *Advances in neural information processing systems*, 32, 2019.

Xiaoming Shi, Siqiao Xue, Kangrui Wang, Fan Zhou, James Zhang, Jun Zhou, Chenhao Tan, and Hongyuan Mei. Language models can improve event prediction by few-shot abductive reasoning. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 29532–29557. Curran Associates, Inc., 2023a. URL `https://proceedings.neurips.cc/paper_files/paper/2023/file/5e5fd18f863cbe6d8ae392a93fd271c9-Paper-Conference.pdf`.

Xiaoming Shi, Siqiao Xue, Kangrui Wang, Fan Zhou, James Y. Zhang, Jun ping Zhou, Chenhao Tan, and Hongyuan Mei. Language models can improve event prediction by few-shot abductive reasoning. *NeurIPS*, abs/2305.16646, 2023b. URL https://api.semanticscholar.org/CorpusID:258947766.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Chris Whong. Foiling nyc's taxi trip data. 2014. URL https://chriswhong.com/open-data/.

Siqiao Xue, Xiaoming Shi, James Zhang, and Hongyuan Mei. Hypro: A hybridly normalized probabilistic model for long-horizon prediction of event sequences. *Advances in Neural Information Processing Systems*, 35:34641–34650, 2022.

Siqiao Xue, Xiaoming Shi, Zhixuan Chu, Yan Wang, Hongyan Hao, Fan Zhou, Caigao Jiang, Chen Pan, James Y Zhang, Qingsong Wen, et al. Easytpp: Towards open benchmarking temporal point processes. *arXiv preprint arXiv:2307.08097*, 2023a.

Siqiao Xue, Yan Wang, Zhixuan Chu, Xiaoming Shi, Caigao Jiang, Hongyan Hao, Gangwei Jiang, Xiaoyun Feng, James Zhang, and Jun Zhou. Prompt-augmented temporal point process for streaming event sequence. *Advances in Neural Information Processing Systems*, 36:18885–18905, 2023b.

Chenghao Yang, Hongyuan Mei, and Jason Eisner. Transformer embeddings of irregularly spaced events and their participants. *ICLR*, abs/2201.00044, 2022. URL https://api.semanticscholar.org/CorpusID:245650405.

Qiang Zhang, Aldo Lipani, Omer Kirnap, and Emine Yilmaz. Self-attentive hawkes process. In *International conference on machine learning*, pp. 11183–11193. PMLR, 2020.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, et al. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*, 2025.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 1(2), 2023.

Feng Zhou, Quyu Kong, Jie Qiao, Cheng Wan, Yixuan Zhang, and Ruichu Cai. Advances in temporal point processes: Bayesian, neural, and llm approaches. *arXiv preprint arXiv:2501.14291*, 2025.

Ke Zhou, Hongyuan Zha, and Le Song. Learning triggering kernels for multi-dimensional hawkes processes. In *International conference on machine learning*, pp. 1301–1309. PMLR, 2013.

Simiao Zuo, Haoming Jiang, Zichong Li, Tuo Zhao, and Hongyuan Zha. Transformer hawkes process. In *International conference on machine learning*, pp. 11692–11702. PMLR, 2020.

# A Related Work

## A.1 Temporal Point Processes (TPPs)

TPPs provide a probabilistic framework for modeling event sequences in continuous time, and recent years have seen a surge of neural TPP models that extend classical approaches with deep learning to capture complex dependencies. The Neural Hawkes Process (NHP) (Mei & Eisner, 2017) models continuous-time dynamics with recurrent neural networks and exponential decay, enabling flexible self-modulating intensities. The Self-Attentive Hawkes Process (SAHP) (Zhang et al., 2020) instead relies on self-attention to encode event histories, improving long-range dependency modeling. Other variants include the Recurrent Marked Temporal Point Process (RMTPP) (Du et al., 2016), which embeds event histories via RNNs for joint time and type prediction; the Fully Neural Network TPP (NNTPP) (Omi et al., 2019), which uses neural parameterizations of conditional distributions for greater expressiveness; and the Transformer Hawkes Process (THP) (Zuo et al., 2020), which combines transformer encoders with point process likelihoods to scale to long sequences.

## A.2 LLMs for Event Modeling

Recent advances have explored the integration of LLMs (Devlin et al., 2019; Zhao et al., 2023) into event prediction, extending beyond traditional TPP. Early efforts such as PromptTPP (Xue et al., 2023b) framed streaming event prediction as a continual learning problem, introducing a retrieval-based prompt pool that enables neural TPPs to adapt to evolving event streams without rehearsal buffers. More recently, works have directly coupled LLMs with temporal dynamics. LAMP (Shi et al., 2023a) demonstrated that LLMs can enhance event forecasting by performing abductive reasoning over candidate futures, thereby improving predictions in few-shot settings. Building on this direction, Language-TPP (Kong et al., 2025) proposed a unified framework that bridges TPPs with LLMs through a novel temporal byte-token encoding, enabling seamless integration of timestamps with textual event information and achieving state-of-the-art results on benchmark datasets. Complementarily, TPP-LLM (Liu & Quan, 2024) leverages parameter-efficient fine-tuning of LLMs combined with temporal embeddings. LAST-SToP (Gupta et al., 2025) explored the use of stochastic soft prompting to adapt pretrained LLMs for asynchronous time series. These studies highlight a growing paradigm shift: rather than treating LLMs merely as text encoders, they can serve as powerful backbones for unified modeling of textual and temporal event dynamics. However, these works generally require fine-tuning of the LLM, which can be computationally expensive.

## A.3 Test-Time Adaptation for Time Series

Few recent studies such as TAFAS (Kim et al., 2025) and its extensions (Grover & Etemad, 2025; Medeiros et al., 2025) propose TTA methods specifically for time series forecasting (TSF), where non-stationarity is a persistent issue. These approaches exploit the partially observed ground truth, which becomes progressively available during inference, to continuously adapt the model without requiring full retraining. In particular, TAFAS introduces lightweight adapter modules that recalibrate inputs and outputs while leaving the backbone architecture untouched, enabling efficient on-the-fly adaptation with minimal overhead. While event prediction is not identical to TSF, both domains share an inherent temporal coherence in their sequential structure. This suggests that principles from TTA in TSF may inspire analogous strategies for event prediction, where models could benefit from exploiting newly observed events to refine future predictions dynamically. However, event prediction also involves dual modalities, i.e. temporal dynamics and event semantics, which makes a direct application of TAFAS infeasible.

# B Additional Experiments

## B.1 Autoregressive sequence prediction

To further illustrate how ETTA behaves in more complex autoregressive settings, we conduct an experiment where the model is tasked to predict an entire event sequence step by step. We initialize each run with the first five events of a sequence as input. The ETTA model first updates its weights using the available observed ground truths, and then generates the next event. Once the ground truth for the newly generated event is revealed, it is added to the prefix, allowing the model to update the weights again before predicting further. This process repeats autoregressively until the full sequence

is generated. At each step, we record the predicted event and compare the prediction with the ground truth. In Figure 3, we present representative visualizations of generated sequences for two randomly sampled sequences from the Amazon dataset. The plots compare the ground truth trajectory with predictions from the frozen SAHP model (BASE), SAHP model with test-time fine-tuning (TTFT), and SAHP model with ETTA. We observe that ETTA enables the model to more closely follow the ground truth trajectory, substantially reducing cumulative error over long horizons compared to both baselines. These results highlight the robustness of adapter-based calibration in mitigating error accumulation and improving stability in long-horizon autoregressive prediction.

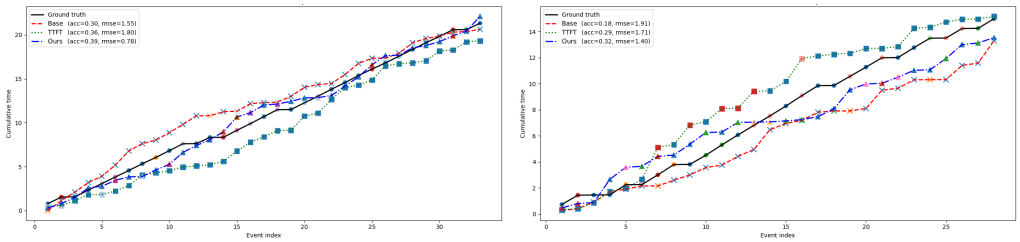


Figure 3: Autoregressive sequence prediction on the Amazon dataset. Starting from the first five observed events, each model predicts the next event, receives the revealed ground truth, and continues iteratively until the full sequence is generated. The plots show cumulative event times for two representative sequences, comparing the frozen SAHP backbone (Base), SAHP with test-time fine-tuning (TTFT), and SAHP with ETTA. ETTA consistently tracks the ground truth more closely, achieving higher accuracy and lower RMSE while reducing error accumulation over long horizons. These results illustrate the robustness of adapter-based calibration for stabilizing autoregressive event prediction.

### B.2 Adapter Ablation Studies

To better understand the role of each adapter in ETTA, we ablate the temporal adapter and the logits adapter, evaluating (i) temporal-only, (ii) logits-only, and (iii) the full dual-adapter ETTA. Tables 4 and 5 show that the two adapters play complementary roles: the temporal adapter primarily improves timestamp prediction (lower RMSE), while the logits adapter primarily improves event-type prediction (higher accuracy). In some cases a single adapter achieves the best value on one metric, but the dual-adapter variant provides the strongest overall trade-off across both accuracy and RMSE.

Table 4: Performance comparison of event prediction using **TPP** backbones on **StackOverflow** and **Taxi** in an ablation study of the temporal vs. logits adapters. We report results for model without adapters (Base), temporal-only (Temp), logits-only (Logits), and for our model with both adapters (ETTA). Higher accuracy and lower RMSE are better; best per dataset/metric is bolded. Results are means over 3 random seeds.

| Backbone | Metric | StackOverflow | | | | Taxi | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | Base | Temp | Logits | ETTA | Base | Temp | Logits | ETTA |
| NHP | Acc. ↑ | 0.4500 | 0.4418 | 0.4599 | **0.4619** | **0.9150** | 0.8918 | 0.9119 | 0.9089 |
| | RMSE ↓ | 1.3790 | 1.3160 | 1.3780 | **1.3120** | 0.3895 | 0.3239 | 0.3798 | **0.3103** |
| SAHP | Acc. ↑ | 0.4390 | 0.4362 | **0.4543** | 0.4533 | 0.9025 | 0.9011 | 0.9155 | **0.9189** |
| | RMSE ↓ | 1.3780 | **1.2840** | 1.3850 | 1.2940 | 0.3506 | 0.3254 | 0.3590 | **0.3098** |
| RMTPP | Acc. ↑ | 0.4270 | 0.4339 | 0.4355 | **0.4388** | 0.9049 | 0.9081 | 0.9103 | **0.9127** |
| | RMSE ↓ | 1.3750 | **1.3290** | 1.3710 | 1.3330 | 0.3790 | **0.3505** | 0.3805 | 0.3510 |
| AttNHP | Acc. ↑ | 0.4480 | 0.4547 | 0.4562 | **0.4612** | 0.9129 | 0.9103 | 0.9123 | **0.9187** |
| | RMSE ↓ | 1.3770 | 1.3190 | 1.3750 | **1.3100** | 0.3780 | 0.3299 | 0.3877 | **0.3270** |

Table 5: Performance comparison of event prediction using **LLM** backbones on **StackOverflow** and **Taxi** in an ablation study of the temporal vs. logits adapters. We report results for model without adapters (Base), temporal-only (Temp), logits-only (Logits), and for our model with both adapters (ETTA). Higher accuracy and lower RMSE are better; best per dataset/metric is bolded. Results are means over 3 random seeds.

| Backbone | Metric | StackOverflow | | | | Taxi | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | Base | Temp | Logits | ETTA | Base | Temp | Logits | ETTA |
| Qwen 3 0.6B | Acc. ↑ | 0.3671 | 0.3609 | 0.3788 | **0.3918** | 0.7510 | 0.7491 | **0.8154** | 0.8121 |
| | RMSE ↓ | 1.4553 | **1.3864** | 1.4425 | 1.3980 | 0.3988 | **0.3566** | 0.3981 | 0.3586 |
| Llama 2 7B | Acc. ↑ | 0.3905 | 0.3912 | **0.4133** | 0.4129 | 0.8022 | 0.8025 | 0.8180 | **0.8195** |
| | RMSE ↓ | **1.4810** | 1.4813 | 1.4992 | 1.4923 | 0.3743 | 0.3687 | 0.3723 | **0.3614** |
| Llama 3 8B | Acc. ↑ | 0.4645 | 0.4599 | **0.4705** | 0.4698 | 0.8819 | 0.8791 | **0.8973** | 0.8950 |
| | RMSE ↓ | 1.4401 | **1.3894** | 1.4512 | 1.3980 | 0.3697 | 0.3551 | 0.3701 | **0.3549** |
| Mistral 7B | Acc. ↑ | 0.3877 | 0.3750 | 0.3992 | **0.4012** | 0.7932 | 0.7917 | **0.8189** | 0.8188 |
| | RMSE ↓ | 1.4180 | 1.3592 | 1.4051 | **1.3564** | 0.3890 | **0.3708** | 0.3887 | 0.3711 |