ReDiTT: Retrieval Augmented Conditional Diffusion Transformers for Asynchronous Time Series

Anonymous authors
Paper under double-blind review

Abstract

We present a diffusion based model for asynchronous time series prediction, where the goal is to predict the next inter event time and event type. To address the inherent uncertainty of future events, we introduce ReDiTT, a retrieval augmented conditional diffusion transformer that operates in latent space. ReDiTT retrieves structurally similar latent sequences from a memory bank during both training and inference and incorporates them as reference conditions through cross attention. This retrieval based conditioning allows the model to attend to relevant temporal dynamics and provides global structural guidance for generation. As a result, ReDiTT stabilizes long horizon forecasting and improves sample diversity. Experiments on seven real world datasets demonstrate state of the art performance on next event prediction and long horizon forecasting.

1 Introduction

Asynchronous time series (*a.k.a.* continuous-time event sequence) prediction arises in a wide range of real world applications, including event driven systems (Enguehard et al., 2020), healthcare monitoring (Lorch et al., 2018; Rizoiu et al., 2018), finance (Bacry et al., 2015; Jin et al., 2020), and user behavior modeling (Hernandez et al., 2017; Zhang et al., 2022; Kong et al., 2023), where observations occur at irregular time intervals rather than on a fixed grid. Unlike regularly sampled time series, asynchronous data encodes information jointly in both event values and inter-event times, leading to complex temporal dynamics that are highly stochastic and nonstationary (Xue et al., 2023). Accurately modeling such data is crucial for downstream tasks such as forecasting, simulation, and decision making, yet remains challenging due to the sparsity, irregularity, and long-range temporal dependencies inherent in these processes (Schirmer et al., 2022; Zhang et al., 2024).

Asynchronous time series prediction is further complicated by the need to model uncertainty and multimodality over future events, particularly in long horizontal forecasting. Classical autoregressive and likelihood-based temporal point process models often rely on strong parametric assumptions or Markovian dynamics (Hawkes, 1971; Mei & Eisner, 2017; Zhang et al., 2020; Zuo et al., 2020; Yang et al., 2021), limiting their ability to generalize to more complex asynchronous time series, particularly those exhibiting rich global structure, long-range dependencies, or mixed continuous and discrete observations.

Recent progress has shown that generative modeling in latent space via variational autoencoders (VAEs) (Higgins et al., 2017) combined with diffusion models (Peebles & Xie, 2023), can effectively capture the stochastic structure of asynchronous time series. In particular, Mukherjee et al. (2025) demonstrates that a VAE can learn a compact and expressive latent representation that supports both accurate reconstruction and diffusion-based next-event and long horizontal forecasting. While diffusion models are powerful generative models, their application to asynchronous time series remains underexplored, despite their ability to generate sequences holistically and mitigate error accumulation compared to autoregressive approaches.

Despite recent progress, unconditional or weakly conditional diffusion models still face significant challenges in long-horizon time series prediction (Liu et al., 2024). Unlike image diffusion models, time series data usually do not come with explicit semantic labels or other strong sources of supervision, which limits the availability of informative global conditioning signals during generation. Without such guidance, the model

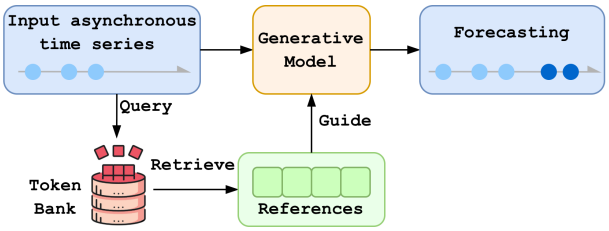

Figure 1: Overview of ReDiTT framework. Given an observed event history, we retrieve the similar sequences from a token memory bank. The retrieved references are injected into diffusion transformer to guide denoising for future forecasting.

has less ability to preserve the specific dynamics of an individual trajectory over extended horizons. As a result, long-term forecasts often drift toward generic, high-probability patterns from the training distribution, rather than maintaining trajectory-specific temporal structure and fine-grained event evolution.

To address these limitations, we propose **ReDiTT** : **Re**trieval Augmented Conditional **Di**ffusion **T**ransformers for Asynchronous **T**ime Series. During training, each sequence retrieves its top k nearest neighbors from a token memory bank, which are then used as conditions. These retrieved sequences share the same latent format as the input and are incorporated through cross-attention modules within the diffusion transformer blocks, allowing the model to explicitly attend to structurally similar temporal dynamics. At inference time, we retrieve the top k references from the token memory bank built from training set to guide generation. An overview of ReDiTT is illustrated in Figure 1. This retrieval-augmented conditioning provides global structural guidance that stabilizes long-horizon forecasting, and improves sample diversity by anchoring generation to concrete examples of temporal dynamics rather than relying purely on learned parameters. Our main contributions are:

① We introduce ReDiTT, the first retrieval-based diffusion framework for asynchronous time series prediction that conditions on top-k retrieved latent priors from a pre-constructed latent token bank.

② We propose a novel conditioning approach for retrieve-based diffusion transformer and show it effectively integrates prior information and provides reasonable guidance.

③ We demonstrate that ReDiTT significantly improves next-event and long-horizon prediction with state-of-the-art performances by experiments on seven real world datasets with a comprehensive analysis.

2 Related Work

2.1 Temporal Point Processes (TPPs)

Marked TPPs are widely adopted as a standard approach for modeling asynchronous time series. A TPP is a stochastic process that generates sequences of discrete events over time. A sequence of n events can be represented as $\mathbf{x} = \{\mathbf{x}_1, \cdots, \mathbf{x}_n\}$, where each event $\mathbf{x}_i = (t_i, e_i)$ consists of the index i indicating the chronological order of events, the inter-event time t_i since the previous event and the corresponding event type e_i. Another widely used variant is to represent events by their inter-event intervals $\tau_i = t_i - t_{i-1}$ rather than the absolute timestamps t_i. These two parameterizations are essentially equivalent, so the literature often switches between them without loss of generality. Classical TPP models (Mei & Eisner, 2017) parameterize conditional intensity functions for next-event prediction and are trained by maximizing the log-likelihood of observed event sequences. Prior work on temporal point processes has largely focused on neural architectures that extend likelihood-based intensity modeling, beginning with RNN-based approaches (Du et al., 2016; Mei & Eisner, 2017) and later incorporating more expressive designs to better capture uncertainty (Mehrasa et al., 2019; Lüdke et al., 2023). More recent Transformer-based TPPs leverage attention mechanisms to improve long range dependency modeling (Zhang et al., 2020; Zuo et al., 2020; Yang et al., 2021), but still rely on

autoregressive intensity formulations. However, the goal has extended beyond forecasting the immediate next event to generating the full future event trajectory, and classical and neural TPPs can be unsatisfactory since errors introduced at early steps accumulate as the autoregressive rollout proceeds. On the optimization side, likelihood-based training typically requires computing distributional quantities implied by the learned intensity, which can become expensive.

2.2 Diffusion Models for Asynchronous Time Series

The shortcomings of classic neural TPP approaches motivate diffusion and flow matching approaches. These methods treat forecasting as conditional generation, transforming simple base noise into future event sequences given the history. By modeling the joint continuation rather than repeatedly sampling one event at a time, diffusion based approaches can better support long-horizon generation and produces diverse futures for uncertainty quantification, offering a strong non-autoregressive alternative to intensity-based neural TPPs. Early diffusion work in time series focused on learning conditional distributions for tasks like imputation (Tashiro et al., 2021) and probabilistic forecasting (Rasul et al., 2021), demonstrating that iterative denoising can capture uncertainty without being constrained to a single greedy rollout. For temporal point processes specifically, Add and Thin (Lüdke et al., 2023) formulates diffusion over full marked event sequences and outperforms autoregressive TPP forecasters. Zhou et al. (2025) embed event sequences into a vector space and diffuse to forecast the full future sequence in one go. Yuan et al. (2023) extended diffusion to spatio-temporal setting. Zeng et al. (2024) coupled two diffusions, one for inter-arrival times and one for event types, to model the joint distribution. More recently, ADiff4TPP (Mukherjee et al., 2025) proposed an asynchronous noise schedule in a VAE latent space to strengthen conditioning via earlier event history when predicting further into the future. While promising, these approaches yield only limited empirical gains, and diffusion for asynchronous event modeling is still relatively understudied. We propose retrieval based diffusion transformers to tackle these challenges and better predict long-horizon future.

2.3 Retrieval Augmented Generation

Retrieval has become a practical way to enhance generative models with non-parametric memory. For text, kNN language models (Khandelwal et al., 2019) augment a base LM by retrieving semantically similar contexts from a data store at inference time. For image, retrieval-augmented diffusion models (Blattmann et al., 2022) condition denoising on retrieved reference images to better capture specific visual structure and long-tail concepts. In time series forecasting, several recent methods (Han et al., 2025; Ning et al., 2025; Tire et al., 2024; Li, 2025) leverage retrieval by selecting historically similar segments and using their continuations as additional context. RATD (Liu et al., 2024) further combines retrieval with diffusion by injecting retrieved references into the denoising process. However, these approaches are primarily designed for regularly sampled time series segments and do not directly apply to asynchronous marked event streams without nontrivial changes. In contrast, our method is tailored to event prediction: we perform retrieval in a VAE latent space that preserves event wise structure, and condition a latent diffusion transformer on retrieved references represented in the same event-sequence format. This design enables trajectory-specific guidance for coherent long horizon event generation, rather than serving only as a segment level short range forecasting booster.

3 Preliminary

3.1 Asynchronous Time Series Forecasting Tasks

Asynchronous time series forecasting is typically evaluated using two tasks. Suppose we observe an event history $\{\mathbf{x}_1, \cdots, \mathbf{x}_i\}$. **Next event prediction** requires a model to forecast the immediate next event $\hat{\mathbf{x}}_{i+1} = (\hat{t}_{i+1}, \hat{e}_{i+1})$, including both its occurrence time or inter-arrival time and its type or mark, conditioned on the observed history. **Long horizon prediction** extends this setting by requiring the model to generate a sequence of future events $\{\hat{\mathbf{x}}_{i+1}, \cdots, \hat{\mathbf{x}}_{i+m}\}$ over a prediction window of length m. This task evaluates the model's ability to capture how uncertainty accumulates as the forecasting horizon increases.

3.2 Flow Matching for Asynchronous Time Series

Flow matching (FM) (Lipman et al., 2023; Liu et al., 2023; Lee et al., 2024; Esser et al., 2024) provides a diffusion style way to train continuous time generative models by directly regressing a velocity field instead of learning scores. Concretely, FM views the generation as solving an ODE that transports a simple base distribution (typically Gaussian noise) to the data distribution via a learned vector field $\frac{d\mathbf{z}_s}{ds} = v_\theta(\mathbf{z}_s, s)$, using an explicit interpolation path $\mathbf{z}_s = a_s\mathbf{z} + b_s\boldsymbol{\epsilon}$, where s denotes a random time $s \sim \mathcal{U}(0,1)$ (we use s to denote the timestep in diffusion model, and t to represent the time value for asynchronous time series data), $\mathbf{z}$ is the input clean sample, and $\boldsymbol{\epsilon} \sim \mathcal{N}(0,1)$ is the Gaussian noise. Rectified flow (Liu et al., 2023; Lee et al., 2024) defines a simpler interpolation $\mathbf{z}_s = (1-s)\mathbf{z} + s\boldsymbol{\epsilon}$. Conditional Flow Matching (CFM) makes training simulation-free by deriving a direct closed-form conditional vector field $u_s(\mathbf{z}_s|\boldsymbol{\epsilon})$ for this path, and learning v_θ by a simple regression loss $\mathbb{E}_{s,\boldsymbol{\epsilon},\mathbf{z}}\big[\|v_\theta(\mathbf{z}_s, s) - u_s(\mathbf{z}_s|\boldsymbol{\epsilon})\|^2\big]$.

To facilitate efficient retrieval and ease the diffusion training, each event $\mathbf{x}_i$ is encoded as a compact latent representation $\mathbf{z}_i \in \mathbb{R}^d$ using a pretrained VAE E_ϕ. Let $\mathbf{x} \in \mathcal{X}$ be an event sequence of length n. We pad $\mathbf{x}$ to a fixed length N, where N is the maximum sequence length in $\mathcal{X}$. The encoder then maps the padded sequence to a latent representation $\mathbf{z} = E_\phi(\mathbf{x}) \in \mathbb{R}^{N\times d}$, which serves as a single training sample for latent flow matching.

To make flow matching respect the causal and unevenly spaced nature of event sequences, asynchronous matrix valued interpolation (Mukherjee et al., 2025) defines:

$$\mathbf{z}_s = \boldsymbol{A}(s)\mathbf{z} + (\boldsymbol{I} - \boldsymbol{A}(s))\boldsymbol{\epsilon}, \tag{1}$$

where $\epsilon = \{\epsilon_1, \cdots, \epsilon_N\}$ is a Gaussian noise of same dimension $N \times d$ as $\mathbf{z}$. And $\boldsymbol{A}(s) \in \mathbb{R}^{N\times N}$ is a diagonal matrix whose per event schedule is designed so that later events are injected with noise earlier (and therefore are trained to be denoised earlier) than earlier events, that being said, later events are corrupted earlier and the model learns to denoise and forecast the tail under stronger noise.

This yields a generative ODE that incorporates the schedule derivative with the chain rule:

$$\frac{d\mathbf{z}_s}{ds} = \boldsymbol{A}'(s)v_\theta(\mathbf{z}_s, \boldsymbol{A}(s)). \tag{2}$$

In each training iteration, we randomly select a time $s \in [0,1]$ and generate the corresponding intermediate state $\mathbf{z}_s$ using Equation (1). Training follows a CFM objective in this asynchronous setting by weighting the regression with $\boldsymbol{A}'(s)$:

$$\mathcal{L}_{\text{CFM}}(\theta) = \mathbb{E}_{s,\boldsymbol{\epsilon},\mathbf{z}}\big[\ \|\boldsymbol{A}'(s)\big(\, v_\theta(\mathbf{z}_s, \boldsymbol{A}(s)) - u_s(\mathbf{z}_s|\boldsymbol{\epsilon})\,\big)\|^2\ \big]. \tag{3}$$

During inference, given a observed history $\mathbf{x}_1, \cdots, \mathbf{x}_n$ and a target prediction horizon size m. We can form a initial condition $\{\mathbf{z}_1, \cdots, \mathbf{z}_n, \epsilon_{n+1}, \cdots, \epsilon_{n+m}\}$ and we can incorporate the history by filling the corresponding entries of $\frac{d\mathbf{z}_s}{ds}$ in Equation (2) with the known latents:

$$\frac{d\mathbf{z}_s}{ds} = \begin{cases} \boldsymbol{A}'(s)_{ii}(\mathbf{z}_i - \epsilon_i) & \text{if } 1 \le i \le n, \\ \boldsymbol{A}'(s)_{ii}v_\theta(\mathbf{z}_s, \boldsymbol{A}(s)) & \text{if } n < i \le m. \end{cases} \tag{4}$$

Standard ODE solvers can solve this without introducing numerical error, which leads us to the predictions $\{\hat{\mathbf{z}}_{n+1}, \cdots, \hat{\mathbf{z}}_{n+m}\}$. The decoder then maps them back to the original space to obtain $\{\hat{\mathbf{x}}_n, \cdots, \hat{\mathbf{x}}_{n+m}\}$.

4 Method

To better ground diffusion-based generation in trajectory-specific dynamics, we propose ReDiTT, a retrieval augmented latent Diffusion Transformer for temporal point processes, as illustrated in Figure 2. Given an observed event history $\{\mathbf{x}_1, \cdots, \mathbf{x}_i\}$ as input, our model generates a future event sequence $\{\hat{\mathbf{x}}_{i+1}, \cdots, \hat{\mathbf{x}}_{i+m}\}$, supporting both next event prediction ($m = 1$) and long horizon prediction ($m > 1$). ReDiTT couples latent space embedding retrieval with conditional generation: we retrieve the top-k most similar historical

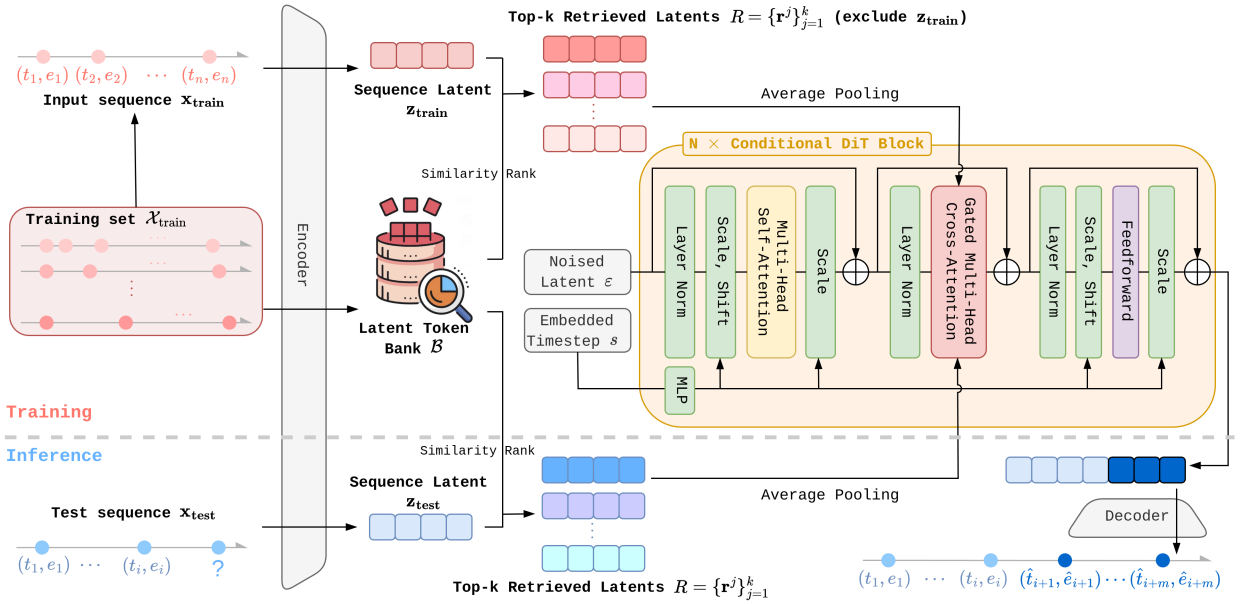

Figure 2: **ReDiTT** models long horizon asynchronous time series forecasting as conditional generation guided by retrieved references. During training, we encode each event sequence $\mathbf{x}_{\text{train}}$ into latent tokens $\mathbf{z}_{\text{train}}$ and build a latent reference bank $\mathcal{B}$. For each sample, we retrieve the top-k nearest neighbors R by latent similarity, aggregate them, and condition a DiT via cross-attention. We then train with the diffusion objective on the future latent segment to denoise and reconstruct clean latents given the prefix and retrieved context. During inference, we encode the observed prefix $\mathbf{x}_{\text{test}} = (t_1, e_1), \ldots, (t_i, e_i)$, apply the same retrieval and aggregation, run the reverse diffusion process to sample future latents, and decode them into event times and types $(\hat{t}_{i+1}, \hat{e}_{i+1}), \cdots, (\hat{t}_{i+m}, \hat{e}_{i+m})$.

trajectories from a reference bank and use them as additional conditioning to guide the diffusion model towards context consistent futures. We first describe our latent space retrieval strategy in Section 4.1. We then introduce the reference guided conditional flow matching objective in Section 4.2. Finally, we detail the conditional DiT architecture that integrates retrieved references via cross-attention in Section 4.3.

4.1 Masked Token Level Retrieval in Latent Space

Directly retrieving similar asynchronous sequences in the original observation space is brittle: each sequence mixes continuous inter-event times with discrete event types, has variable length with padding, and can admit multiple valid alignments (e.g. two sequences can be the same pattern but shifted in time, events can be permuted locally). As a result, naive distances (e.g., Euclidean over concatenated features) are easily dominated by scale mismatch, sparsity, or padding artifacts, and often fail to reflect trajectory-level similarity. To obtain a more reliable notion of neighborhood, we perform retrieval in the latent space of the pretrained VAE E_ϕ, where each event $\mathbf{x}_i = (t_i, e_i)$ is mapped to a compact representation $\mathbf{z}_i \in \mathbb{R}^d$ that, by construction, captures the global dynamics needed for reconstruction. Operating in latent space allows the diffusion model to consume conditioning signals in the same representation space as the generation target. This alignment makes it easier to model the joint distribution of event sequence latents and to use retrieved references as direct guidance for coherent multi-event forecasting. Consider an asynchronous sequence $\mathbf{x} = \{\mathbf{x}_1, \cdots, \mathbf{x}_N\}$ with a padding mask $\mathbf{m} \in \{0, 1\}^N$ (this mask is common in data processing to keep a constant total event length among all the sequence), we use E_ϕ to produce per-event latent tokens $\mathbf{z} = E_\phi(\mathbf{x}) \in \mathbb{R}^{N \times d}$. In addition, we precompute a latent token bank $\mathcal{B} = \{(\mathbf{z} = E_\phi(\mathbf{x}), \mathbf{m}) | \, \mathbf{x} \in \mathcal{X}_{\text{train}}\}$ for the training database. We then define a masked cosine similarity between a query $(\boldsymbol{q}, \mathbf{m}^q) \in E_\phi(\mathcal{X})$ and a

reference candidate $(\mathbf{r}, \mathbf{m}^r) \in \mathcal{B}$ as a token-wise cosine averaged over valid time steps:

$$\texttt{sim}(\boldsymbol{q}, \mathbf{r}) = \frac{1}{\sum_i \mathbf{1}[\mathbf{m}_i^q \wedge \mathbf{m}_i^r]} \sum_{i:\mathbf{m}_i^q \wedge \mathbf{m}_i^r} \frac{\langle \hat{\mathbf{q}}_i, \hat{\mathbf{r}}_i \rangle}{\|\hat{\mathbf{q}}_i\| \|\hat{\mathbf{r}}_i\|}, \tag{5}$$

where $\hat{q}, \hat{r}$ denote ℓ_2 normalized tokens. Finally, we retrieve the top k references for $\boldsymbol{q}$:

$$R(\boldsymbol{q}) = \{\mathbf{r} \,|\, \mathrm{TopK}_{r \in \mathcal{B}} \, \texttt{sim}(\boldsymbol{q}, \mathbf{r}), \mathbf{r} \neq \boldsymbol{q}\}, \tag{6}$$

which will enter the diffusion process as extra condition. During training, we exclude the closest reference, since it is the query sample $\boldsymbol{q}$ itself and would dominate the conditioning signal. Instead, we retrieve from the second closest onward, which improves generalization at inference time.

4.2 Conditional Flow Matching with Retrieved References

Let R denote the external reference condition associated with a target latent event sequence $\mathbf{z}$. Conditioning does not alter the flow matching construction and Equation (1) still holds, but we now learn a conditional vector field $v_\theta(\cdot \,|\, R)$ that transports samples along this path while being guided by R. The conditional flow matching objective becomes:

$$\mathcal{L}_{\mathrm{CFM}}(\theta) = \mathbb{E}_{s,\boldsymbol{\epsilon},\mathbf{z}}\big[\, \| \boldsymbol{A}'(s) \big(v_\theta(\mathbf{z}_s, \boldsymbol{A}(s) \,|\, R) - u_s(\mathbf{z}_s | \boldsymbol{\epsilon}) \big) \|^2 \, \big]. \tag{7}$$

Intuitively, R does not modify the geometry of the bridge from $\boldsymbol{\epsilon}$ to $\mathbf{z}$, but it provides additional information that biases the learned dynamics toward trajectory-consistent solutions, which is especially helpful for long-horizon generation.

During training, we retrieve k references R from the same training bank $\mathcal{B}$ and feed the latent tokens as an addition condition in every DiT block, encouraging the conditional flow to model a set of plausible futures consistent with similar historical trajectories. During inference, we retrieve from the same training bank again , so the sequences that model has already encountered can serve as an in-distribution prototype that stabilizes long-horizon generation and reduces drift when extrapolating far beyond the local context.

4.3 Conditional Diffusion Transformer Architecture

We modify the Transformer based DiT blocks because they have the following issues when applying to asynchronous time series: conditioning is often injected through adaptive normalization (e.g., AdaLN) where a global conditioning vector (typically built from the diffusion timestep and a sparse label) modulates each block. For asynchronous sequences, this kind of conditioning is intrinsically limited: the timestep carries no instance-specific semantics, and event "labels" (if any) are usually coarse or unavailable, so the conditioning signal is weak and globally broadcast to all tokens in the same way.

After retrieving k nearest-neighbor reference sequences in latent space, we therefore construct a sequence-shaped condition that mirrors the input format. Say we have retrieve $R = \{\mathbf{r}^j\}_{j=1}^k$ for input latent $\mathbf{z}$, we first obtain a reference sequence by weighted pooling:

$$\mathbf{r} = \sum_{j=1}^{k} w_j \mathbf{r}^j \in \mathbb{R}^{N \times d}, \;\; w_j = \texttt{softmax}\big(\texttt{sim}(\mathbf{z}, \mathbf{r}^j)\big). \tag{8}$$

Before entering the DiT blocks, both current latent input and retrieved reference are expanded to hidden size D by channel repetition, for $\mathbf{r}$, we also broadcast the mask $\mathbf{m}^{\mathbf{z}}$ across the channel dimension. Then $\mathbf{z}$ and $\mathbf{r}$ are augmented with fixed positional embeddings $\texttt{pos}$ to get $\mathbf{z} = \texttt{Repeat}(\mathbf{z}) + \texttt{pos}_{\mathbf{z}} \in \mathbb{R}^{N \times D}$ and $\mathbf{r} = \texttt{Repeat}(\mathbf{r}) + \texttt{pos}_{\mathbf{r}} \in \mathbb{R}^{N \times D}$. Considering the reference has the same format as the input, it is naturally to inject $\mathbf{r}$ via cross-attention after self attention in each DiT block as illustrated in Figure 2. After the LayerNorm, the module forms:

$$\boldsymbol{Q} = \boldsymbol{W}_q(\mathbf{LN}(\mathbf{z})), \;\; \boldsymbol{K} = \boldsymbol{W}_k(\mathbf{LN}(\mathbf{r})), \;\; \boldsymbol{V} = \boldsymbol{W}_v(\mathbf{LN}(\mathbf{r})), \tag{9}$$

splits into α heads, and computes masked attention weights over the flattened memory positions:

$$\mathbf{Attn} = \texttt{MaskedSoftmax}(\frac{\boldsymbol{Q}\boldsymbol{K}^{\intercal}}{\sqrt{D/\alpha}}; \mathbf{m}^{r}). \tag{10}$$

The attention output is $\boldsymbol{Y} = \mathbf{Attn}\boldsymbol{V}$, followed by an output projection $\boldsymbol{W}_{\text{out}}$ and a learned scalar sigmoid gate $c \in \mathbb{R}$:

$$\mathbf{z} = \mathbf{z} + \sigma(c)\boldsymbol{W}_{\text{out}}\boldsymbol{Y}. \tag{11}$$

Our cross-attention conditioning is effective as it injects the retrieved reference as a token level external memory, letting each latent token selectively attend to the most relevant reference positions rather than relying on a coarse global conditioning vector. The sigmoid gated residual makes this guidance stable and adaptive, i.e. starting weak and strengthening only when it improves generation, so the model can leverage retrieval without overwhelming the base dynamics. More details are discussed in Appendix Section B.3.

5 Experiments

5.1 Experimental Setup

Datasets. Following Xue et al. (2023), we run experiments on five standard temporal point process datasets: Amazon (Ni et al., 2019) with 16 event types; Retweet (Zhou et al., 2013) with 3 event types; StackOverflow (Leskovec & Krevl, 2014) with 22 event types; Taobao (Xue et al., 2022) with 20 event types; Taxi (Whong, 2014) with 10 event types. We also evaluate ReDiTT on two action datasets: Breakfast (Kuehne et al., 2014) with 177 action classes, we scale the timestamp by dividing by 100 to avoid VAE training loss explosion; Multithumos (Yeung et al., 2018) with 65 action classes, we take the 212 sequences with length ≤ 100 out of total 289 sequences to avoid model ran out of memory. More details are in Appendix Section B.1 and Table 5.

Baselines. We compare ReDiTT with four kinds of state-of-the-art TPP models for asynchronous time series: (1) RNN-based models, including RMTPP (Du et al., 2016) and NHP (Mei & Eisner, 2017); (2) Attention-based models, including SAHP (Zhang et al., 2020), THP (Zuo et al., 2020), AttNHP (Yang et al., 2021), and DTPP (Panos, 2024); (3) Diffusion-based models, including Add&Thin (Lüdke et al., 2023) and ADiff4TPP (Mukherjee et al., 2025); (4) Other popular models including IFTPP (Shchur et al., 2020) and HYPRO (Xue et al., 2022).

Metrics. Following Xue et al. (2023), we examine our models on next event prediction. We evaluate the time prediction by root mean square error (RMSE) between predicted time and true time, and the event type prediction by accuracy between predicted type and true type. Note to compute the accumulated RMSE and accuracy, when predicting $\hat{\mathbf{x}}_{i+2}$, the model uses the true history $\{\mathbf{x}_1, \cdots, \mathbf{x}_i, \mathbf{x}_{i+1}\}$ rather than a previously predicted event $\{\mathbf{x}_1, \cdots, \mathbf{x}_i, \hat{\mathbf{x}}_{i+1}\}$. We also examine ReDiTT on long-horizon prediction by computing the optimal transport distance (OTD), a measurement of edit distance between the predicted $\{\hat{\mathbf{x}}_{i+1}, \cdots, \hat{\mathbf{x}}_{i+m}\}$ and the ground truth $\{\mathbf{x}_{i+1}, \cdots, \mathbf{x}_{i+m}\}$. We follow the implementation of OTD by Mei et al. (2019) with dynamic programming to efficiently find alignment and compute distance between predictions and true events. Following Mukherjee et al. (2025), we compute the OTD with horizon window size $m = 5, 10, 20, 30$ respectively.

To facilitate reproducibility, we provide full implementation details and hyperparameter settings in Appendix Section B.

5.2 Next-event Prediction and Long Horizon Prediction

In this section, we evaluate our model against existing methods on next-event prediction and long-horizon prediction tasks. Overall, **ReDiTT outperforms existing baselines on both tasks and is particularly effective in regimes with long event sequences and limited training samples**, where purely parametric models struggle to capture rare transitions and long-range temporal dependencies. By retrieving trajectory-level neighbors as explicit context, ReDiTT provides strong guidance for coherent long-horizon

Table 1: **Next-event prediction and long-horizon prediction results** for ReDiTT and existing baselines on seven benchmark datasets. RMSE is computed on the predicted inter-event time, while Accuracy (↑) is computed on the predicted event type. Long-horizon forecasting performance is measured by OTD (↓) at horizons $m = 5, 10, 20, 30$. We use ADiff4TPP as the unconditional diffusion baseline and include additional baselines provided by Xue et al. (2023). When available, we report the results published in Mukherjee et al. (2025) for direct comparison. The **best results** are highlighted in bold, and the second best results are underlined. The numbers in grey within parentheses indicate the standard deviation with three random seeds 1,2,3.

	Amazon		Retweet		StackOverflow		Taobao		Taxi		Breakfast		MultiThumos	
	RMSE (↓)	Acc% (↑)	RMSE (↓)	Acc% (↑)	RMSE (↓)	Acc% (↑)	RMSE (↓)	Acc% (↑)	RMSE (↓)	Acc% (↑)	RMSE (↓)	Acc% (↑)	RMSE (↓)	Acc% (↑)
RMTPP	0.559 (0.014)	29.6 (0.008)	26.207 (5.620)	51.6 (0.030)	1.246 (0.293)	42.4 (0.002)	0.257 (0.073)	43.6 (0.000)	0.351 (0.042)	88.5 (0.002)	1.080 (0.393)	4.8 (0.001)	7.357 (0.021)	0.1 (0.000)
NHP	0.640 (0.002)	30.0 (0.001)	22.511 (0.033)	53.9 (0.001)	1.324 (0.359)	26.8 (0.146)	0.168 (0.098)	49.3 (0.012)	0.342 (0.075)	87 (0.007)	1.168 (0.050)	4.9 (0.001)	7.309 (0.005)	9.9 (0.002)
SAHP	0.517 (0.008)	32.0 (0.005)	21.708 (0.001)	54.0 (0.001)	1.327 (0.002)	42.3 (0.002)	0.154 (0.083)	46.4 (0.009)	0.335 (0.175)	88.1 (0.016)	1.130 (0.002)	4.8 (0.000)	7.674 (0.102)	10.3 (0.018)
THP	0.550 (0.016)	34.6 (0.002)	26.176 (0.059)	59.5 (0.001)	1.424 (0.012)	42.0 (0.013)	0.314 (0.044)	45.6 (0.017)	0.375 (0.065)	87.3 (0.011)	1.134 (0.052)	1.3 (0.000)	9.254 (0.625)	13.0 (0.016)
AttNHP	0.755 (0.185)	31.9 (0.011)	22.296 (1.135)	57.2 (0.041)	1.350 (0.018)	44.6 (0.001)	0.280 (0.085)	47.1 (0.019)	0.429 (0.012)	85.2 (0.027)	1.123 (0.027)	4.7 (0.001)	7.887 (0.094)	14.9 (0.013)
IFTPP	0.465 (0.001)	35.1 (0.001)	22.198 (0.234)	60.0 (0.002)	1.884 (0.043)	45.5 (0.008)	0.598 (0.103)	55.9 (0.003)	0.357 (0.013)	91.4 (0.004)	3.654 (0.134)	9.2 (0.003)	8.980 (0.092)	14.3 (0.014)
DTPP	0.619 (0.104)	34.5 (0.002)	24.680 (6.364)	59.7 (0.003)	1.780 (0.167)	39.3 (0.002)	0.587 (0.031)	46.7 (0.003)	0.302 (0.043)	87.9 (0.012)	-	-	-	-
Add&Thin	0.461 (0.017)	-	22.914 (0.348)	-	1.469 (0.238)	-	0.440 (0.035)	-	0.368 (0.015)	-	-	-	-	-
HYPRO	0.583 (0.012)	33.8 (0.004)	20.562 (1.633)	60.0 (0.049)	1.417 (0.253)	44.9 (0.002)	0.307 (0.029)	55.1 (0.004)	0.383 (0.008)	86.5 (0.018)	-	-	-	-
Adiff4TPP	0.413 (0.005)	33.7 (0.003)	17.480 (0.023)	60.7 (0.001)	1.524 (0.003)	34.8 (0.013)	0.140 (0.054)	57.4 (0.011)	0.309 (0.003)	85.6 (0.029)	1.360 (0.006)	8.2 (0.001)	5.981 (0.003)	16.3 (0.000)
ReDiTT (k=1)	0.410 (0.004)	49.5 (0.004)	15.238 (0.081)	69.6 (0.012)	1.240 (0.001)	41.9 (0.005)	**0.134** (0.002)	58.8 (0.008)	0.253 (0.002)	92.8 (0.009)	1.210 (0.003)	10.8 (0.002)	5.800 (0.002)	17.2 (0.000)
ReDiTT (k=7)	**0.352** (0.002)	**60.9** (0.000)	**13.429** (0.020)	**78.5** (0.005)	**1.048** (0.001)	**53.0** (0.002)	0.140 (0.000)	**59.3** (0.001)	**0.239** (0.002)	**94.5** (0.001)	**1.054** (0.015)	**15.1** (0.000)	**4.797** (0.102)	**25.0** (0.000)
	OTD (↓)		OTD (↓)		OTD (↓)		OTD (↓)		OTD (↓)		OTD (↓)		OTD (↓)	
RMTPP	9.9 / 19.7 / 39.2 / 58.4 (0.016 / 0.035 / 0.075 / 0.120)		10.0 / 21.0 / 40.0 / 59.8 (0.003 / 0.004 / 0.012 / 0.021)		9.5 / 18.7 / 37.0 / 54.8 (0.075 / 0.160 / 0.320 / 0.487)		8.7 / 15.9 / 28.7 / 40.6 (0.097 / 0.196 / 0.506 / 0.746)		4.0 / 6.0 / 9.2 / 12.2 (0.136 / 0.196 / 0.326 / 0.449)		9.8 / 19.2 / 34.2 / 45.5 (0.076 / 0.204 / 0.712 / 0.808)		10.0 / 20.0 / 36.8 / 49.0 (0.002 / 0.002 / 0.502 / 0.614)	
NHP	9.9 / 19.7 / 39.2 / 58.4 (0.015 / 0.035 / 0.075 / 0.119)		10.0 / 20.0 / 39.7 / 59.3 (0.003 / 0.008 / 0.028 / 0.056)		9.5 / 18.8 / 37.2 / 55.2 (0.073 / 0.156 / 0.306 / 0.462)		8.5 / 15.2 / 28.1 / 39.3 (0.099 / 0.248 / 0.523 / 0.766)		4.0 / 5.9 / 9.2 / 12.1 (0.137 / 0.199 / 0.327 / 0.452)		9.8 / 10.2 / 34.0 / 45.2 (0.103 / 0.210 / 0.692 / 0.845)		9.9 / 19.7 / 35.9 / 47.7 (0.095 / 0.198 / 0.492 / 0.781)	
SAHP	9.9 / 19.7 / 39.1 / 58.3 (0.016 / 0.035 / 0.080 / 0.124)		10.0 / 20.0 / 39.9 / 59.6 (0.002 / 0.006 / 0.020 / 0.038)		9.5 / 18.8 / 37.2 / 55.2 (0.072 / 0.156 / 0.304 / 0.459)		8.6 / 15.5 / 28.3 / 39.9 (0.101 / 0.247 / 0.523 / 0.735)		5.5 / 7.6 / 12.8 / 17.5 (0.113 / 0.162 / 0.231 / 0.413)		10.0 / 19.5 / 34.7 / 46.1 (0.002 / 0.196 / 0.625 / 0.817)		10.0 / 20.0 / 36.7 / 48.8 (0.002 / 0.001 / 0.712 / 0.975)	
THP	9.9 / 19.7 / 39.1 / 58.2 (0.017 / 0.037 / 0.082 / 0.130)		10.0 / 20.0 / 39.7 / 59.2 (0.003 / 010 / 0.032 / 0.063)		9.5 / 18.8 / 37.1 / 54.9 (0.069 / 0.148 / 0.306 / 0.469)		8.7 / 15.8 / 28.8 / 40.7 (0.095 / 0.234 / 0.499 / 0.735)		6.0 / 10.8 / 14.3 / 17.6 (0.094 / 0.163 / 0.318 / 0.388)		10.0 / 19.6 / 34.9 / 46.3 (0.001 / 0.419 / 0.518 / 0.800)		9.7 / 19.4 / 35.6 / 46.9 (0.132 / 0.593 / 0.613 / 0.913)	
AttNHP	9.9 / 19.7 / 39.1 / 58.2 (0.018 / 0.039 / 0.083 / 0.130)		10.0 / 20.0 / 39.7 / 59.2 (0.005 / 0.012 / 0.032 / 0.065)		9.4 / 18.7 / 36.9 / 54.4 (0.073 / 0.157 / 0.323 / 0.492)		8.2 / 14.7 / 24.0 / 38.4 (0.105 / 0.247 / 0.499 / 0.723)		4.0 / 5.8 / 9.0 / 11.7 (0.105 / 0.247 / 0.240 / 0.307)		9.8 / 19.2 / 34.0 / 45.1 (0.102 / 0.419 / 0.592 / 0.710)		9.9 / 19.9 / 35.7 / 47.2 (0.094 / 0.258 / 0.540 / 0.714)	
IFTPP	9.9 / 19.7 / 40.0 / 58.1 (0.018 / 0.042 / 0.091 / 0.140)		10.0 / 20.0 / 39.7 / 59.2 (0.003 / 0.011 / 0.031 / 0.064)		9.4 / 18.6 / 36.8 / 54.3 (0.077 / 0.164 / 0.331 / 0.500)		8.0 / 14.2 / 26.2 / 37.6 (0.104 / 0.245 / 0.503 / 0.703)		4.5 / 5.6 / 8.6 / 11.6 (0.098 / 0.162 / 0.221 / 0.337)		8.8 / 16.7 / 28.5 / 37.7 (0.009 / 0.219 / 0.493 / 0.712)		9.9 / 19.6 / 35.0 / 46.4 (0.104 / 0.278 / 0.515 / 0.74)	
DTPP	6.9 / 13.7 / 27.6 / 41.7 (0.010 / 0.024 / 0.064 / 0.161)		10.0 / 19.4 / 29.8 / 39.6 (0.004 / 0.009 / 0.003 / 0.063)		7.6 / 15.0 / 29.2 / 43.4 (0.020 / 0.036 / 0.036 / 0.486)		6.8 / 15.1 / 32.3 / 50.0 (0.019 / 0.032 / 0.136 / 0.684)		3.0 / 6.8 / 13.8 / 20.7 (0.013 / 0.019 / 0.145 / 0.482)		-		-	
HYPRO	7.0 / 13.0 / 26.1 / 34.1 (0.016 / 0.055 / 0.140 / 0.155)		9.0 / 19.7 / 30.7 / 39.0 (0.003 / 0.005 / 0.063 / 0.037)		7.3 / 12.2 / 29.2 / 36.8 (0.017 / 0.028 / 0.454 / 0.563)		5.8 / 11.4 / 20.8 / 30.7 (0.028 / 0.047 / 0.074 / 0.198)		3.4 / 5.8 / 10.0 / 13.9 (0.023 / 0.034 / 0.056 / 0.072)		-		-	
Adiff4TPP	6.2 / 12.4 / 24.7 / 32.9 (0.049 / 0.115 / 0.141 / 0.477)		9.1 / 17.7 / 28.0 / 31.7 (0.017 / 0.051 / 0.170 / 0.208)		6.5 / 12.0 / 22.3 / 30.1 (0.400 / 0.922 / 2.142 / 1.789)		4.9 / 9.9 / 20.4 / 31.1 (0.086 / 0.065 / 0.149 / 0.131)		**2.4 / 4.0 / 6.8 / 9.4** (0.022 / 0.015 / 0.049 / 0.012)		8.8 / 17.1 / 24.3 / 28.9 (0.090 / 0.102 / 0.139 / 0.158)		8.2 / 13.6 / 21.1 / 20.2 (0.092 / 0.132 / 0.156 / 0.198)	
ReDiTT (k=1)	5.9 / 11.5 / 21.3 / 27.8 (0.02 / 0.103 / 0.110 / 0.329)		9.1 / 17.3 / 26.9 / 30.1 (0.016 / 0.020 / 0.167 / 0.107)		6.1 / 10.8 / 19.4 / 27.4 (0.89 / 0.893 / 1.210 / 1.623)		4.8 / 9.8 / 19.8 / 30.7 (0.021 / 0.080 / 0.128 / 0.129)		2.6 / 4.4 / 7.9 / 12.7 (0.027 / 0.192 / 0.111 / 0.912)		8.7 / 16.1 / 22.1 / 27.0 (0.05 / 0.182 / 0.135 / 0.300)		8.2 / 14.3 / 20.6 / 19.6 (0.02 / 0.112 / 0.167 / 0.175)	
ReDiTT (k=7)	**5.7 / 10.7 / 19.7 / 25.8** (0.041 / 0.100 / 0.140 / 0.301)		**9.1 / 17.2 / 26.5 / 29.6** (0.018 / 0.042 / 0.112 / 0.124)		**6.0 / 10.7 / 18.9 / 27.4** (0.291 / 0.420 / 0.910 / 1.394)		**4.6 / 9.4 / 19.4 / 29.5** (0.031 / 0.086 / 0.159 / 0.171)		3.5 / 5.0 / 9.2 / 17.6 (0.005 / 0.209 / 0.468 / 0.923)		**8.2 / 15.4 / 21.6 / 26.4** (0.005 / 0.100 / 0.142 / 0.179)		**8.2 / 14.1 / 20.6 / 19.6** (0.002 / 0.032 / 0.095 / 0.124)	

generation under sparse supervision. Beyond retrieval itself, the results demonstrate that the manner in which retrieved information is injected into the model is critical. The proposed conditioning strategy, based on cross attention in diffusion transformers, enables the denoising network to selectively attend to the most relevant retrieved context, resulting in more accurate and coherent predictions.

Next-event prediction. Table 1 summarizes the main results on seven benchmark datasets. Performance is evaluated on next-event prediction, using RMSE for time prediction and accuracy for type prediction. Overall, ReDiTT exhibits the strongest and most consistent performance across a broad range of dynamics. It achieves the best RMSE and the highest type accuracy on all datasets. ReDiTT substantially outperforms both classic neural TPP baselines and recent diffusion-based or flow-based forecasters. In addition, ReDiTT delivers large gains in average performance across datasets compared to the strongest baseline, ADiff4TPP (Mukherjee et al., 2025). The average RMSE is reduced by 22%, decreasing from 3.887 for ADiff4TPP to 3.008 for ReDiTT. At the same time, the average type accuracy increases by 12 percentage points, from 42.4% to 55.2%. The advantages of ReDiTT are especially pronounced on Breakfast and MultiThumos, which feature large event vocabularies but substantially fewer training sequences. In this low-data, high-cardinality regime, purely parametric models are prone to sparse supervision for rare event types. Retrieval provides valuable exemplar-based guidance by exposing the model to structurally similar trajectories and richer type co-occurrence patterns. On Breakfast, which contains 177 event types, ReDiTT improves type accuracy from 8.2% to 15.1%. A consistent trend is observed on MultiThumos with 65 event types, where

ReDiTT yields a large accuracy gain from 16.3% to 25.0%. These results indicate that retrieval is particularly effective when the label space is large and training coverage per class is limited.

Long horizon prediction. For long-horizon prediction, ReDiTT consistently reduces OTD at horizons $m = 5, 10, 20, 30$ on six datasets compared to the strongest diffusion baseline (Mukherjee et al., 2025). This result demonstrates that retrieval guidance substantially improves sequence-level coherence beyond single-step prediction. The gains are most pronounced on Amazon, where ReDiTT lowers OTD from 12.4 to 10.7 at horizon 10 and from 32.9 to 25.8 at horizon 30, indicating that the advantages of retrieval-augmented conditioning grow as the prediction horizon increases. Similar patterns are observed on Retweet and StackOverflow, where ReDiTT consistently achieves the best OTD across horizons, highlighting the robustness of the approach under diverse temporal dynamics. These long-horizon improvements are driven by the retrieval-conditional design, which injects trajectory-relevant training examples as explicit context. This design enables the model to generate multiple future events coherently while preserving temporal ordering and capturing long-range sequential structure. On Taxi, ReDiTT achieves the best RMSE and accuracy, while OTD is optimized at smaller retrieval sizes. This behavior aligns with the dataset characteristics, which include short sequences and a limited event vocabulary. In this setting, retrieval provides smaller incremental benefits because sequence-level alignment is already well constrained. Overall, these findings support the core motivation of retrieval-augmented conditioning. By injecting trajectory-relevant training examples as explicit context, the model is more effectively guided toward plausible future dynamics.

5.3 Ablation Studies

Different Conditioning Architecture. As discussed in Peebles & Xie (2023), adaptive layer norm often outperforms cross-attention conditioning in diffusion transformers for class-labeled image generation. We therefore conduct an ablation on five datasets with fixed $k = 1$ to analyze which conditioning architecture better suits asynchronous event sequences. Following the design in Peebles & Xie (2023), we implement adaLN conditioning on $\mathbf{r}$ and also consider $\text{Concat}(s, \mathbf{r})$, which concatenates the diffusion timestamp s with $\mathbf{r}$ to encode temporal information, i.e. three paths for s, $\mathbf{r}$ and the fused $\text{Concat}(s, \mathbf{r})$, each is controlled by a scale hyperparameter. More details can be found in Appendix Section B.3. As reported in Table 2, cross-attention consistently yields stronger long-horizon behavior, achieving substantially lower OTD, whereas adaLN only improves event-type accuracy on three of the five datasets and often degrades sequence-level alignment. We observe a similar pattern: any gains adaLN provides in next-event RMSE or type accuracy are relatively modest, while its long-horizon OTD deteriorates substantially. This indicates that adaLN's improvements are largely limited to short-term, local prediction, whereas cross-attention is far more reliable for preserving sequence-level alignment over extended horizons.

We attribute this difference to the mismatch between the inductive bias of adaLN and the structure of conditioning signals in asynchronous time series. In class-labeled image generation, the condition is typically a single global discrete cue shared across the entire sample; in this setting, adaLN's global, layer-wise modulation can effectively inject the label signal and amplify class conditional priors. In contrast, for asynchronous time series, retrieved input-like references are high-entropy and event-dependent, with information that varies across events and time. Cross-attention therefore provides a more suitable mechanism by enabling state-dependent access to context and selectively integrating temporal patterns as the generated history evolves. Moreover, adaLN is also more expensive in our implementation, resulting in a 294M parameter model, whereas the cross-attention variant is substantially lighter with only 209M parameters. Cross-attention also converges faster in practice, reaching strong performance in fewer training iterations. To obtain both predictive quality and computational efficiency, we adopt cross-attention conditioning in our final model.

Different Aggregation of Retrieved References. We further compare two strategies for incorporating the top k retrieved references during conditioning: average pooling and concatenation. We perform the experiments on StackOverflow dataset and the results are illustrated in Table 3. Under a fixed retrieval budget k, we observe that average pooling of the retrieved references consistently outperforms concatenation across three metrics. Moreover, as k increases, the performance gap widens, mostly notably on event-type prediction accuracy. For concatenation, the accuracy at $k = 7$ is lower than at $k = 5$. We hypothesize that pooling

Table 2: Analysis of different conditioning architecture of DiT for five datasets with a fixed $k = 1$. For unconditional training, we refer to ADiff4TPP.

	condition	RMSE (↓)	Acc% (↑)	OTD (↓)
Amazon	uncondition	0.413	33.7	6.2 / 12.4 / 24.7 / 32.9
	adaLN	0.460	**51.0**	7.7 / 15.2 / 28.7 / 36.1
	crossattn	**0.410**	49.5	**5.9 / 11.5 / 21.3 / 27.8**
Retweet	uncondition	17.480	60.7	9.1 / 17.7 / 28.0 / 31.7
	adaLN	19.266	69.4	9.6 / 18.9 / 30.1 / 33.9
	crossattn	**15.238**	**69.6**	**9.1 / 17.3 / 26.9 / 30.1**
StackOverflow	uncondition	1.524	34.8	6.5 / 12.0 / 22.3 / 30.1
	adaLN	**1.012**	**51.3**	9.4 / 18.8 / 37.3 / 55.5
	crossattn	1.240	41.9	**6.1 / 10.8 / 19.4 / 27.4**
Taobao	uncondition	0.140	57.4	4.9 / 9.9 / 20.4 / 31.1
	adaLN	0.268	**60.5**	5.6 / 10.9 / 22.1 / 32.2
	crossattn	**0.134**	58.8	**4.8 / 9.8 / 19.8 / 30.7**
Taxi	uncondition	0.309	85.6	**2.4 / 4.0 / 6.8 / 9.4**
	adaLN	0.263	**94.0**	6.8 / 11.8 / 23.4 / 34.7
	crossattn	**0.253**	92.8	2.6 / 4.4 / 7.9 / 12.7

Table 3: Analysis of the different aggregation strategy and different k for retrieved references on StackOverflow. Overall, increasing k leads to improved performance, and average pooling outperforms concatenation as k increases.

		RMSE (↓)	Acc% (↑)	OTD (↓)
uncondition		1.524	34.8	6.5 / 12.0 / 22.3 / 30.1
$k = 1$		1.240	41.9	6.1 / 10.8 / 19.4 / 27.4
concatenation	$k = 3$	1.244	42.1	6.2 / 10.9 / 19.6 / 27.8
	$k = 5$	1.174	47.1	6.1 / 10.9 / 19.7 / 27.9
	$k = 7$	1.163	44.9	6.0 / 11.1 / 19.7 / 27.6
average pooling	$k = 3$	1.126	48.7	6.0 / 10.7 / 19.2 / 27.8
	$k = 5$	1.101	50.6	6.0 / 10.7 / 19.4 / 28.8
	$k = 7$	**1.048**	**53.0**	**6.0 / 10.7 / 18.9 / 27.4**

becomes increasingly beneficial with larger retrieved sets because it acts as a robust aggregation operator, that being said, it emphasizes signals that are consistent across references with high similarity weights while dampening irrelevant information. In contrast, concatenation scales the conditioning length linearly with k, which increases exposure to irrelevant retrievals and makes it easier for the model to overfit to noisy reference. The results showed that this effect becomes more obvious as k grows, leading to a larger degradation in categorical prediction. We therefore adopt average pooling in our conditioning implementation.

Different k for Retrieval Mechanism. We also conducted an ablation study on the number of retrieved references k to further investigate how the amount of retrieved context influences conditioning effectiveness while keeping all other settings fixed, specifically, whether a small set of references provides sufficient guidance for generation, or whether increasing k introduces irrelevant matches that act as noise and degrade performance. As listed in Table 3, larger k does not improves the results for concatenation conditioning much. However, for average pooling conditioning, increasing k consistently improves both RMSE and event type accuracy, whereas OTD remains largely stable across different values of k. This suggests that retrieving more neighbors can provide additional contextual signal that helps the model refine local predictions, particularly for event type inference, while preserving the global sequence level structure captured by OTD. Notably, event type accuracy reaches 53% at $k = 7$, representing a 10.9 point improvement over $k = 1$. This indicates that multi-reference conditioning is beneficial, and a moderate retrieval size can yield significant gains, especially for categorical prediction, while preserving long-horizon sequence alignment.

Effect of Time and Type Conditioning Components. We further ablate the conditioning signal by using time-only or event-type-only guidance. Because retrieval is performed in the latent space, we first retrieve reference sequences via latent similarity, and then mask out the undesired modality by replacing the corresponding latent block (time or type) with a zero latent (i.e., the latent obtained by encoding an all-zero

Table 4: Analysis of different conditioning components with a fixed $k = 1$ on Taobao. Overall, conditioning on both event time and event type outperforms conditioning on either time or type alone.

	RMSE (↓)	Acc% (↑)	OTD (↓)
time & type	**0.134**	58.5	**4.8 / 9.8 / 19.8 / 30.7**
time only	0.135	59.1	4.7 / 9.9 / 20.3 / 31.2
type only	0.134	**61.1**	4.9 / 10.2 / 20.9 / 31.5

input in the original space), while keeping the other block unchanged. This design isolates the contribution of each modality without altering the retrieval set.

We evaluate this ablation on Taobao (20 event types) with fixed $k = 1$. As shown in Table 4, relative to using both modalities, time-only conditioning yields slightly better OTD for short-horizon prediction (window of 5), but slightly worse OTD at longer horizons (windows of 20 and 30), suggesting that temporal cues alone can guide near-term dynamics but are insufficient to maintain event semantics over extended rollouts. In contrast, type-only conditioning substantially improves event-type accuracy, but consistently degrades OTD across all horizons, indicating that categorical guidance alone can sharpen local classification while hurting sequence-level alignment.

6 Conclusion

We propose ReDiTT, a retrieval augmented conditional Diffusion Transformer for asynchronous event sequence forecasting that models the joint uncertainty of the next inter event time and event type. ReDiTT operates in latent space and uses a memory bank to retrieve structurally similar latent sequences as reference conditions during training and inference. The retrieved references are injected through cross attention in DiT blocks, enabling the model to selectively use trajectory specific temporal patterns that are hard to capture with purely parametric conditioning. Across seven real world datasets, ReDiTT achieves state-of-the-art results on both next event prediction and long horizon forecasting. Ablations confirm the value of retrieval based conditioning and cross attention integration, and show that structured guidance is critical for stable long horizon generation in temporal point processes.

References

Emmanuel Bacry, Iacopo Mastromatteo, and Jean-François Muzy. Hawkes processes in finance. *Market Microstructure and Liquidity*, 1(01):1550005, 2015.

Andreas Blattmann, Robin Rombach, Kaan Oktay, Jonas Müller, and Björn Ommer. Retrieval-augmented diffusion models. *Advances in Neural Information Processing Systems*, 35:15309–15324, 2022.

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.

Joseph Enguehard, Dan Busbridge, Adam Bozson, Claire Woodcock, and Nils Hammerla. Neural temporal point processes for modelling electronic health records. In *Machine Learning for Health*, pp. 85–113. PMLR, 2020.

Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *Forty-first international conference on machine learning*, 2024.

Sungwon Han, Seungeon Lee, Meeyoung Cha, Sercan O Arik, and Jinsung Yoon. Retrieval augmented time series forecasting. In *Forty-second International Conference on Machine Learning*, 2025.

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

Sergio Hernandez, Pedro Alvarez, Javier Fabra, and Joaquin Ezpeleta. Analysis of users' behavior in structured e-commerce websites. *IEEE Access*, 5:11941–11958, 2017.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *International conference on learning representations*, 2017.

Zhuochen Jin, Shunan Guo, Nan Chen, Daniel Weiskopf, David Gotz, and Nan Cao. Visual causality analysis of event sequence data. *IEEE transactions on visualization and computer graphics*, 27(2):1343–1352, 2020.

Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. Generalization through memorization: Nearest neighbor language models. *arXiv preprint arXiv:1911.00172*, 2019.

Quyu Kong, Pio Calderon, Rohit Ram, Olga Boichak, and Marian-Andrei Rizoiu. Interval-censored transformer hawkes: Detecting information operations using the reaction of social systems. In *Proceedings of the ACM web conference 2023*, pp. 1813–1821, 2023.

Hilde Kuehne, Ali Arslan, and Thomas Serre. The language of actions: Recovering the syntax and semantics of goal-directed human activities. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 780–787, 2014.

Sangyun Lee, Zinan Lin, and Giulia Fanti. Improving the training of rectified flows. *Advances in neural information processing systems*, 37:63082–63109, 2024.

Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data, June 2014.

Zichao Li. Retrieval-augmented forecasting with tabular time series data. In *Proceedings of the 4th Table Representation Learning Workshop*, pp. 192–199, 2025.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023.

Jingwei Liu, Ling Yang, Hongyan Li, and Shenda Hong. Retrieval-augmented diffusion models for time series forecasting. *Advances in Neural Information Processing Systems*, 37:2766–2786, 2024.

Xingchao Liu, Chengyue Gong, et al. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *The Eleventh International Conference on Learning Representations*, 2023.

Lars Lorch, Abir De, Samir Bhatt, William Trouleau, Utkarsh Upadhyay, and Manuel Gomez-Rodriguez. Stochastic optimal control of epidemic processes in networks. *arXiv preprint arXiv:1810.13043*, 2018.

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, 2023.

Nazanin Mehrasa, Akash Abdu Jyothi, Thibaut Durand, Jiawei He, Leonid Sigal, and Greg Mori. A variational auto-encoder model for stochastic point processes. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 3165–3174, 2019.

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.

Hongyuan Mei, Guanghui Qin, and Jason Eisner. Imputing missing events in continuous-time event streams. In *International Conference on Machine Learning*, pp. 4475–4485. PMLR, 2019.

Amartya Mukherjee, Ruizhi Deng, He Zhao, Yuzhen Mao, Leonid Sigal, and Frederick Tung. ADiff4TPP: Asynchronous Diffusion Models for Temporal Point Processes. *arXiv 2504.20411*, 2025.

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.

Kanghui Ning, Zijie Pan, Yu Liu, Yushan Jiang, James Yiming Zhang, Kashif Rasul, Anderson Schneider, Lintao Ma, Yuriy Nevmyvaka, and Dongjin Song. Ts-rag: Retrieval-augmented generation based time series foundation models are stronger zero-shot forecaster. *arXiv preprint arXiv:2503.07649*, 2025.

Aristeidis Panos. Decomposable transformer point processes. *Advances in Neural Information Processing Systems*, 37:88932–88955, 2024.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 4195–4205, 2023.

Kashif Rasul, Calvin Seward, Ingmar Schuster, and Roland Vollgraf. Autoregressive denoising diffusion models for multivariate probabilistic time series forecasting. In *International conference on machine learning*, pp. 8857–8868. PMLR, 2021.

Marian-Andrei Rizoiu, Swapnil Mishra, Quyu Kong, Mark Carman, and Lexing Xie. Sir-hawkes: on the relationship between epidemic models and hawkes point processes. *The Web Confernce*, 2018.

Mona Schirmer, Mazin Eltayeb, Stefan Lessmann, and Maja Rudolph. Modeling irregular time series with continuous recurrent units. In *International conference on machine learning*, pp. 19388–19405. PMLR, 2022.

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

Yusuke Tashiro, Jiaming Song, Yang Song, and Stefano Ermon. Csdi: Conditional score-based diffusion models for probabilistic time series imputation. *Advances in neural information processing systems*, 34: 24804–24816, 2021.

Kutay Tire, Ege Onur Taga, Muhammed Emrullah Ildiz, and Samet Oymak. Retrieval augmented time series forecasting. *arXiv preprint arXiv:2411.08249*, 2024.

Chris Whong. Nyc taxi open data. https://chriswhong.com/open-data/foil_nyc_taxi/, 2014.

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*, 2023.

Chenghao Yang, Hongyuan Mei, and Jason Eisner. Transformer embeddings of irregularly spaced events and their participants. *arXiv preprint arXiv:2201.00044*, 2021.

Serena Yeung, Olga Russakovsky, Ning Jin, Mykhaylo Andriluka, Greg Mori, and Li Fei-Fei. Every moment counts: Dense detailed labeling of actions in complex videos. *International Journal of Computer Vision*, 126(2):375–389, 2018.

Yuan Yuan, Jingtao Ding, Chenyang Shao, Depeng Jin, and Yong Li. Spatio-temporal diffusion point processes. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 3173–3184, 2023.

Mai Zeng, Florence Regol, and Mark Coates. Interacting diffusion processes for event sequence forecasting. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 58407–58430, 2024.

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

Weijia Zhang, Chenlong Yin, Hao Liu, Xiaofang Zhou, and Hui Xiong. Irregular multivariate time series forecasting: A transformable patching graph neural networks approach. In *Forty-first International Conference on Machine Learning*, 2024.

Yizhou Zhang, Defu Cao, and Yan Liu. Counterfactual neural temporal point process for estimating causal influence of misinformation on social media. *Advances in Neural Information Processing Systems*, 35: 10643–10655, 2022.

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.

Wang-Tao Zhou, Zhao Kang, Ling Tian, Jinchuan Zhang, and Yumeng Liu. Non-autoregressive diffusion-based temporal point processes for continuous-time long-term event prediction. *Expert Systems with Applications*, 267:126210, 2025.

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 Limitations and Future Work

Limitations. Although ReDiTT achieves substantial empirical gains, it introduces additional system overhead by requiring a reference bank and performing retrieval at inference time, which increases both memory usage and latency. Another limitation is that gains can be limited on small-scale datasets, especially when the sequence length is relatively short, the model might have limited improvement on capturing long range behaviors.

Future Work. Future directions include learning retrieval representations that are better aligned with forecasting objectives, and making conditioning more robust via reference weighting or retrieval dropout. On the efficiency side, we can reduce cost with compressed or hierarchical reference banks and caching. It is also promising to incorporate richer context such as covariates or text, and to explore training objectives that more directly target long-horizon metrics.

B Implementation Details

B.1 Dataset Preparation and Baselines

Following (Xue et al., 2023), we pad Amazon, Retweet, StackOverflow, Taobao, and Taxi to the maximum sequence length within each dataset. For Breakfast, we rescale time values by dividing by 100 to stabilize VAE training. For the two text-based action datasets, Breakfast and MultiThumos, we encode event types as integer indices in the same way as the other datasets, and split the data into train/validation/test sets with a 70/10/20 ratio. Additional dataset details are provided in dataset settings of Table 5.

For baseline models, we use the implementations from EasyTPP (Xue et al., 2023) for RMTPP Du et al. (2016), NHP Mei & Eisner (2017), SAHP Zhang et al. (2020), THP Zuo et al. (2020), AttNHP Yang et al. (2021), and IFTPP Shchur et al. (2020).

For other baselines, we adapt DTPP Panos (2024) to predict inter-event times in their original scale rather than in log space, following Adiff4TPP Mukherjee et al. (2025). For Add&Thin Lüdke et al. (2023), the method outputs only inter-event times, so we do not report event-type accuracy or OTD.

B.2 VAE Training

Our VAE implementation for asynchronous time series follows Adiff4TPP Mukherjee et al. (2025). For each event $\mathbf{x} = (t, e)$, we train a β-VAE consisting of an encoder E_ϕ that maps $\mathbf{x}$ to a latent variable $\mathbf{z}$, modeled as

a Gaussian distribution parameterized by an inferred mean (and variance). A decoder D_ϕ then reconstructs the event from $\mathbf{z}$, producing $\tilde{\mathbf{x}} = (\tilde{t}, \tilde{e}) = D_\phi(\mathbf{z})$. The whole β-VAE is optimized with the following objective:

$$\mathcal{L} = (t - \tilde{t})^2 + \mathrm{CE}(e, \tilde{e}) + \beta \mathcal{L}_{KL}, \tag{12}$$

β controls the weight of the KL regularization term, which penalizes deviations of the latent distribution $\mathcal{X}$ from a standard Gaussian prior. As shown in the VAE settings of Table 5, we use the same β choices reported in the appendix of Mukherjee et al. (2025).

B.3 DiT Implementation

Adapting DiT for Asynchronous Time Series. Following Peebles & Xie (2023), we instantiate $v_\theta(\cdot, \boldsymbol{A}(s))$ with a Diffusion Transformer (DiT) backbone, since its Transformer blocks provide a flexible way to model long-range dependencies while remaining compatible with diffusion or flow-style training objectives. We only make minimal modifications to adapt DiT from images to asynchronous event sequences. In particular, we remove the image-specific patchify and patch-embedding modules and instead feed the model with latent event tokens produced by the encoder E_ϕ. This replacement preserves the core DiT computation (stacked Transformer blocks with conditioning) while ensuring that the model operates directly on a sequence of event-level representations, which is the natural input format for asynchronous time series forecasting.

When initializing the model, we specify a hyperparameter N that sets the maximum sequence length the model can represent and generate. Concretely, sequences shorter than N are padded and masked, allowing us to use a fixed token length for efficient batched training and inference while still supporting variable-length histories and horizons. Finally, to inject the conditioning matrix $\boldsymbol{A}(s) \in \mathbb{R}^{N \times N}$, we extend the original DiT conditioning design by constructing an embedding that reflects both temporal progression and structural interactions encoded by $\boldsymbol{A}(s)$. We achieve this by broadcasting the entries of $\boldsymbol{A}(s)$ through sinusoidal frequency features, producing a dense embedding that provides the Transformer blocks with a smooth, scale-aware representation of the matrix. This design allows the network to exploit the temporal and structural dynamics captured by $\boldsymbol{A}(s)$ without introducing additional architectural complexity.

Timestep Embedding. We set the maximum period to $T_{\max} = 10000$ and use an embedding horizon of $f = 128$ sinusodial frequencies. Given the diagonal elements of $\boldsymbol{A}(s) \in \mathbb{R}^{N \times N}$, we construct a frequency-scaled argument matrix $\boldsymbol{B} \in \mathbb{R}^{N \times h}$ to encode each event position with multiple time scales. Concretely, for $i \in \{1, \cdots, N\}$ and $j \in \{1, \cdots, h\}$, we define:

$$\boldsymbol{B}_{ij} = \boldsymbol{A}(s)_{ii} \cdot T_{\max}^{-\frac{j-1}{f}} \tag{13}$$

so that j indexes a geometric progression of frequencies spanning from coarse to fine resolutions. We then form the final timestep embedding $\mathtt{emb}_s \in \mathbb{R}^{N \times 2f}$ by concatenating sine and cosine features:

$$\mathtt{emb}_s = [\cos(\boldsymbol{B}), \sin(\boldsymbol{B})]. \tag{14}$$

Then we map this to the model hidden size D via an MLP:

$$\mathtt{emb}_s = \boldsymbol{W}_2 \mathbf{SiLU}(\boldsymbol{W}_1 \mathtt{emb}_s + b_1) + b_2 \in \mathbb{R}^{N \times D} \tag{15}$$

Following the choice of Mukherjee et al. (2025), we define the schedule function $\boldsymbol{A}$ as:

$$\boldsymbol{A}(s)_{ij} = \begin{cases} 0 & \text{if } i \neq j, \\ \max(0, \min(\frac{2N - i - s(2N-1)}{N}, 1)) & \text{if } i = j, \end{cases} \tag{16}$$

which is a diagonal matrix that assigns a separate noise schedule to each event position. The schedule is constructed so that tokens corresponding to later events receive noise at earlier diffusion times than tokens

for earlier events. Equivalently, later events are corrupted sooner, and the model is trained to denoise them earlier during the reverse process, forcing it to learn forecasting of the sequence tail under higher noise levels.

This sinusoidal construction provides a smooth multi-scale representation of the time conditioning signal $\boldsymbol{A}(s)$, allowing the Transformer blocks to access both low frequency (global) and high frequency (local) temporal variations through a fixed dimension embedding.

Positional Embedding. Before entering the DiT blocks, both current latent input and retrieved reference are expanded to hidden size D by channel repetition, for $\mathbf{r}$, we also broadcast the mask $\mathbf{m}^z$ across the channel dimension. Then $\mathbf{z}$ and $\mathbf{r}$ are augmented with fixed positional embeddings $\texttt{pos}$ to get:

$$\mathbf{z} = \text{Repeat}(\mathbf{z}) + \texttt{pos}_{\mathbf{z}} \in \mathbb{R}^{N\times D},\ \mathbf{r} = \text{Repeat}(\mathbf{r}) + \texttt{pos}_{\mathbf{r}} \in \mathbb{R}^{N\times D}, \tag{17}$$

where $\texttt{pos}$ is defined as follows. For input sequence $\mathbf{z}$ with N being $\texttt{num\_rows}$, we assign each position an integer "row index" $p \in \{0, 1, \cdots, N-1\}$. For embedding dimension D, define frequency coefficients:

$$\gamma_i = 10000^{-\frac{i}{D/2}}, i = 0, \cdots, \frac{D}{2} - 1. \tag{18}$$

The fixed positional embedding for row p is:

$$\texttt{pos}(p) = [\sin(p\gamma), \cos(p\gamma)] \in \mathbb{R}^D. \tag{19}$$

Stacking all rows gives:

$$\texttt{pos} \in \mathbb{R}^{N\times D} \tag{20}$$

Since our retrieved reference $\mathbf{r}$ and input $\mathbf{z}$ has the same dimension, so we apply the same $\texttt{pos}$ embedding to both of them.

Mask Mechanism. When training or sampling sequences whose length n is smaller than the maximum token budget N, we apply an attention mask within multi-head self-attention and the multi-head cross attention. so that padded positions do not participate in the computation. The similar strategy is designed in the retrieval process to mask the padded positions. This masking enforces that the Transformer operates only on valid event tokens, preserving the natural ordering of the sequence and preventing information leakage through padding. Combined with our asynchronous noise schedule, the masked attention mechanism aligns the model's computation with the underlying temporal structure of the data. It also makes the framework compatible with variable-length inputs and flexible forecasting horizons, since the same architecture can condition on any observed prefix and generate different prediction window sizes without changing the model.

AdaLN Conditioning Implementation. We also include the details of our adaLN conditioning for retrieved asynchronous time series. Inside the DiT block, let $\mathbf{z} \in \mathbb{R}^{N\times D}$ be the token features entering the block, we have three conditioning streams:

- time embedding $\texttt{emb}_s$,
- reference embedding $\mathbf{r}$,
- fused conditioning $\texttt{Concat}(s, \mathbf{r})$.

We first apply separate LayerNorms and project the concatenation to the desired dimension:

$$s = \mathbf{LN}_s(s),\ \mathbf{r} = \mathbf{LN}_{\mathbf{r}}(\mathbf{r}),\ \text{cond} = \texttt{Proj}(\texttt{Concat}(s, \mathbf{r})) = \boldsymbol{W}_{\text{cond}}[s; \mathbf{r}] + b_{\text{cond}} \in \mathbb{R}^{N\times D}, \tag{21}$$

where $[\,\cdot\,;\,\cdot\,]$ concatenates along channels and $\boldsymbol{W}_{\text{cond}} \in \mathbb{R}^{D\times 2D}$. Each stream produces a modulation vector via an MLP:

$$\boldsymbol{M}_s = f_s(s) \in \mathbb{R}^{N\times 6D},\ \boldsymbol{M}_{\mathbf{r}} = f_{\mathbf{r}}(\mathbf{r}) \in \mathbb{R}^{N\times 6D},\ \boldsymbol{M}_{\text{cond}} = f_{\text{cond}}(\text{cond}) \in \mathbb{R}^{N\times 6D}. \tag{22}$$

Then we scale each modulator by a learned scalar passed through a sigmoid:

$$\boldsymbol{M}_s = \sigma(\lambda_s)\boldsymbol{M}_s,\ \boldsymbol{M}_{\mathbf{r}} = \sigma(\lambda_{\mathbf{r}})\boldsymbol{M}_{\mathbf{r}},\ \boldsymbol{M}_{\text{cond}} = \sigma(\lambda_{\text{cond}})\boldsymbol{M}_{\text{cond}}, \tag{23}$$

with trainable $\lambda_s, \lambda_{\mathbf{r}}, \lambda_{\text{cond}} \in \mathbb{R}$.

The combined modulation is:

$$\boldsymbol{M} = \boldsymbol{M}_s + \boldsymbol{M}_{\mathbf{r}} + a \cdot \boldsymbol{M}_{\text{cond}} \in \mathbb{R}^{N\times 6D}, \text{ where } a \in \{0,1\} \text{ is a flag hyperparameter.} \tag{24}$$

Then $\boldsymbol{M}$ is split into six D-dimension tensors and enters the layers as vanilla adaLN in DiT.

Multi-reference Cross-attention by Concatenating k References into Length kN memory. The main query is $\mathbf{z} \in \mathbb{R}^{N\times D}$, and in the average pooling case $\mathbf{r} \in \mathbb{R}^{N\times D}$, we just need to adapt to $\mathbf{r} \in \mathbb{R}^{N\times k\times D}$ with per reference masks $\mathbf{m} \in \{0,1\}^{k\times N}$. Similarly to Section 4.3, the only changes are $\boldsymbol{K} \in \mathbb{R}^{N\times k\times D}, \boldsymbol{V} \in \mathbb{R}^{N\times k\times D}$, then split them into α heads with $\boldsymbol{K} \in \mathbb{R}^{\alpha\times(kN)\times\frac{D}{\alpha}}, \boldsymbol{V} \in \mathbb{R}^{\alpha\times(kN)\times\frac{D}{\alpha}}$.

B.4 Hyperparameters and Computation Cost

We report the dataset specifications, model hyperparameters, and computation costs in Table 5.

The long time reported for next-event prediction mainly comes from the autoregressive evaluation protocol over the full sequence length N. Following prior work (Xue et al., 2023; Mukherjee et al., 2025), we compute this metric by iterating $i = 1, \cdots, N-1$ and predicting the next event $\hat{\mathbf{x}}_{i+1}$ conditioned on the prefix $\{\mathbf{x}_1, \cdots, \mathbf{x}_i\}$. Consequently, obtaining the full set of one-step-ahead predictions $\hat{\mathbf{x}}_2, \cdots, \hat{\mathbf{x}}_N$ requires $N-1$ separate forward passes, after which RMSE and accuracy are evaluated between $\hat{\mathbf{x}}$ and $\mathbf{x}$. The detailed algorithm is provided in Appendix E of Mukherjee et al. (2025). We use the implementation from Mukherjee et al. (2025), and this implementation is not optimized for inference; therefore, the reported time could be reduced with further implementation improvements. For a single sequence, ReDiTT takes 2.1s for one-step next-event prediction, compared to 2.0s for the unconditional baseline of Mukherjee et al. (2025). This indicates that the retrieval-conditioned design adds only a reasonable runtime overhead. Because next-event prediction is relatively slow under the standard autoregressive evaluation protocol, this also motivates our long-horizon prediction setting: instead of repeatedly running one-step inference many times, we generate an entire future segment in a single run, enabling more efficient forecasting of long sequences.

We also include the iteration numbers we used in ablation tables in Table 6.

Table 5: ReDiTT Hyperparameters of reported results in Table 1. For the next-event prediction task, inference time is reported using an auto-regressive methodology.

		Amazon	Retweet	StackOverflow	Taobao	Taxi	Breakfast	Multithumos
dataset	# event types	16	3	22	20	10	177	64
	max length	94	97	101	64	38	131	100
	# train data	6454	9000	1401	1300	1400	179	209
	# test data	1851	1520	401	500	400	52	61
	# validation data	922	1535	401	200	200	25	27
VAE	latent dimension d	32	32	32	32	32	32	32
	$\beta_{\max}$	1e-2	1e-1	1e-2	1e-2	1e-2	1e-2	1e-2
DiT	latent size	96	96	96	96	96	96	96
	hidden size	1152	1152	1152	1152	1152	1152	1152
	depth	7	7	7	7	7	7	7
	# heads	16	16	16	16	16	16	16
	mlp ratio	4	4	4	4	4	4	4
train	batch size	4	4	4	4	4	4	4
	iterations for k=1	500k	550k	550k	30k	950k	100k	100k
	iterations for k=3	-	-	500k	-	-	-	-
	iterations for k=5	-	-	500k	-	-	-	-
	iterations for k=7	450k	450k	550k	25k	50k	400k	350k
	lr	3e-5	3e-5	3e-5	3e-5	3e-5	3e-5	3e-5
	ema decay	0.9999	0.9999	0.9999	0.9999	0.9999	0.9999	0.9999
	time per 50k iteration	50 min	50 min	50min	50min	50min	50min	50min
test	total time for next-event prediction	18h12min	16h38min	5h	1h30min	14min	1h20min	43min
	total time for long-horizon prediction	1h35min	1h24min	24min	11min	3min	5min	4min
GPU		1 A100 40G	1 A100 40G	1 A100 40G	1 A100 40G	1 A100 40G	1 A100 40G	1 A100 40G

Table 6: To facilitate reproducibility, we report the training iteration counts associated with the results presented in Table 2, Table 3, and Table 4 respectively.

	Amazon	Retweet	StackOverflow	Taobao	Taxi
uncondition	950k	950k	900k	150k	900k
adaLN	950k	950k	950k	950k	950k
crossattn	550k	550k	550k	30k	950k

	k=1	k=3	k=5	k=7
concatenation	550k	600k	750k	550k
average pooling	550k	500k	500k	550k

time & type	30k
time only	50k
type only	50k