TIME SERIES FOUNDATION MODELS CAN BE EFFECTIVE FORECASTERS FOR IRREGULAR TIME SERIES

Anonymous authors
Paper under double-blind review

ABSTRACT

Time series foundation models (TSFMs) offer strong zero-shot forecasting across domains. Their ability to transfer without task-specific training makes them attractive for sparse irregular forecasting, yet they are typically applied to regularly sampled sequences. Inspired by temporal point process modeling, we study an inference-time event-sequence interface that represents each observed event by its value and inter-event interval, enabling frozen TSFMs to produce next-event value and timing forecasts without regular-grid imputation. On 85 TimeIMM series from nine dataset families, the event-sequence interface achieves lower normalized value MAE and MSE than regular-grid interfaces for each of the five evaluated TSFMs; both errors are also lower in 8 of 9 dataset families. Both interfaces use the same frozen TSFM checkpoint and are evaluated on the same held-out examples, providing evidence that the input interface can be an important performance bottleneck for frozen TSFMs on irregular time series. Compared with four trained irregular baselines, frozen TSFMs using the event-sequence interface achieve lower next-event value and timing errors. On seven EasyTPP datasets, frozen TSFMs using the event-sequence interface provide competitive next-event timing forecasts relative to TPP baselines trained on each dataset. Together, the results show that frozen TSFMs can effectively forecast sparse irregular time series through the event-sequence interface. Our code is at `https://anonymous.4open.science/r/tsfm_sparse-F81B`.

1 INTRODUCTION

Time series foundation models (TSFMs) have rapidly improved zero-shot forecasting on dense, regularly sampled sequences. Recent systems such as Chronos-2, CTSM 1.0, TimesFM 2.5, and Toto 2.0 show that large pretrained forecasters can transfer across domains without task-specific training (Ansari et al., 2025; Gou et al., 2025; Das et al., 2024; Khwaja et al., 2026). This progress, however, is still largely mediated through a regular-grid interface: a model receives values at ordered positions on a dense sequence and forecasts values at future sequence positions. Many operational, medical, network, and human-behavior time series do not naturally arrive this way because observations may be event-triggered, constrained by data collection, or disrupted by missingness and delays (Chang et al., 2025). They are sparse and irregular, with information concentrated at observed events rather than at every point on a fixed grid. This mismatch raises a basic input-interface question: how should irregular observations be presented to a pretrained forecaster whose native input is an ordered sequence?[1]

To use a regular-sequence forecaster on such data, the irregular timeline is typically converted into ordered grid positions (Shukla & Marlin, 2020; Chang et al., 2025). As illustrated by Panels A–B of fig. 1, this conversion places the sparse observations on a regular grid. Common preprocessing choices include sample-and-hold, linear interpolation, zero fill, and masks that distinguish observed from imputed grid values (Che et al., 2018). These are reasonable engineering choices, but they also change the input-output problem seen by the model. Observed events are mixed with synthetic grid values, long intervals with no observations occupy sequence positions, and the forecasting horizon to the next recorded event varies with the number of intervening artificial grid positions. These effects

[1] We use input interface to mean the input sequence, variable representation, and forecast target used when applying an otherwise frozen TSFM.

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

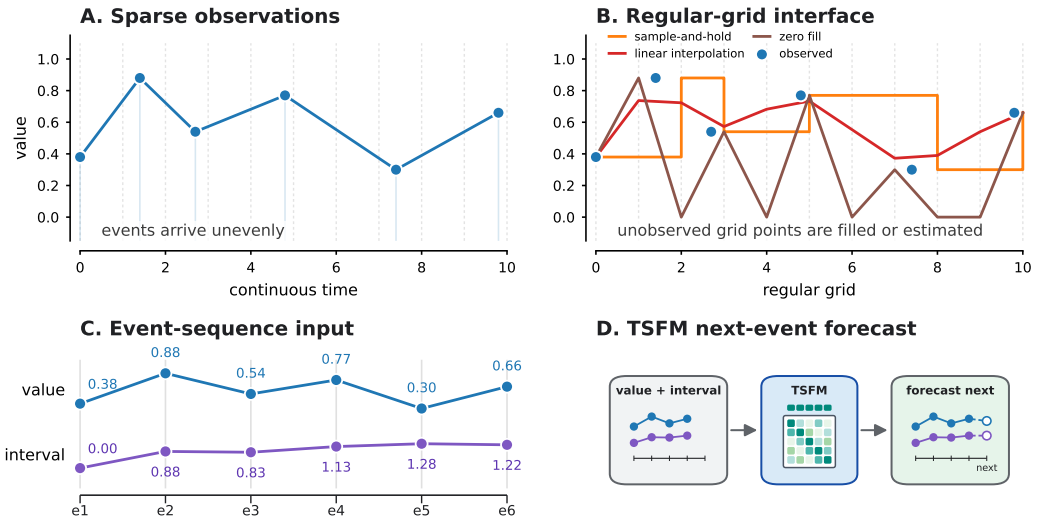


Figure 1: Panels A–B contrast sparse observations with regular-grid interfaces that turn irregular observations into dense sequences with artificial values. Panels C–D show the event-sequence TSFM interface: observed events are kept in order and irregularity is exposed through an inter-event interval dimension $\phi(\Delta t)$, allowing the TSFM to produce a next-event value forecast and a next-event timing point forecast.

are not failures of any particular filling, interpolation, or masking method; they are consequences of forcing sparse event data into a dense regular-grid interface.

These representation effects matter especially when TSFMs are used zero-shot, because their frozen parameters cannot adapt to the chosen preprocessing pipeline. The input interface therefore carries more of the burden. It determines which facts about the sparse history are visible, which positions consume context, and what forecast horizon the model must cover. We therefore study an inference-time event-sequence interface that repurposes released TSFMs for sparse irregular forecasting. It keeps observed events as sequence positions and exposes irregularity through an inter-event interval dimension.

Concretely, let observed values $\mathbf{x}_1, \ldots, \mathbf{x}_n$ arrive at event times $t_1, \ldots, t_n$. Define $\Delta t_1 = 0$ and $\Delta t_i = t_i - t_{i-1}$ for $i > 1$. Using the standard inter-event interval representation, we construct an ordered input sequence whose elements contain the observed value and an interval dimension $\phi(\Delta t_i)$. This mirrors the inter-event interval representation used in temporal point processes (TPP), but here the interval is used as a numeric TSFM input/output coordinate rather than as a likelihood model. At future event-order positions, the TSFM forecasts values and inter-event intervals; future timestamps are recovered by cumulatively summing the predicted intervals from the last observed time. Depending on the model, the value and interval dimensions are forecast either jointly as a multivariate series or separately as univariate event-ordered series. This is a lightweight input transformation for existing models: it requires no additional pretraining, fine-tuning, architectural modification, or continuous-time inference routine. It lets us compare two ways of presenting the same irregular history to a TSFM: a regular grid makes future clock ticks the forecast positions, whereas the event sequence makes future observed events the forecast positions.

The event-sequence interface can be understood as a reindexing of the forecasting problem. A frozen TSFM supplies forecasts over ordered numeric sequence positions. For a regular-grid interface, those positions are clock ticks. For a sparse irregular series, the observed events themselves provide another ordered axis. By constructing event-sequence element $\mathbf{u}_i = [\mathbf{x}_i, r_i]$, where $r_i = \phi(\Delta t_i)$, the next sequence position $\mathbf{u}_{i+1}$ corresponds to the next observed event, and a valid predicted interval coordinate $\hat{r}_{i+1}$ gives $\hat{t}_{i+1} = t_i + \phi^{-1}(\hat{r}_{i+1})$. The interface therefore places next-event value and timing prediction inside the continuation format used by TSFMs, while avoiding synthetic grid

positions that do not correspond to observed events. The reported experiments decode the TSFM interval output into $\hat{t}_{i+1}$ and report it as a next-event timing point forecast. This yields a conditional next-event point-forecast formulation in which the continuation target is the next observed event.

Our primary evaluation uses Time-IMM (Chang et al., 2025) for a next-event forecasting task on irregular time series. Given an event history, frozen TSFMs using the event-sequence interface predict both the next observed value and the timing of the next event. Across 85 Time-IMM series from nine dataset families, the event-sequence interface gives lower normalized value MAE and MSE than regular-grid interfaces for all five evaluated TSFMs. At the dataset-family level, the event-sequence interface outperforms regular-grid interfaces in 8/9 families for both value MAE and value MSE. Frozen TSFMs using the event-sequence interface also outperform the trained irregular baselines when those baselines jointly predict timing in 9/9 families for value MAE, 8/9 for value MSE, and 9/9 for both timing MAE and timing MSE. Even when trained baselines are given the true next-event timestamp and scored only on value, their best value errors are still worse than the best errors achieved by frozen TSFMs using the event-sequence interface (0.606 vs. 0.353 MAE and 0.800 vs. 0.470 MSE). These Time-IMM results provide evidence that the input interface can be an important bottleneck when frozen TSFMs are applied to irregular time series. We further evaluate next-event timing prediction on seven EasyTPP datasets with categorical event labels (Xue et al., 2024); frozen TSFMs using the event-sequence interface provide competitive timing point forecasts relative to TPP baselines trained on each dataset. Together, the results show that sparse observations can be exposed to frozen TSFMs as event sequences, providing a practical input interface for sparse irregular forecasting tasks.

Our contributions are:

- We empirically demonstrate that the input interface can be an important performance bottleneck when frozen TSFMs are applied to sparse irregular forecasting. We introduce an inference-time event-sequence interface that represents observed events as sequence positions and timing as an inter-event interval dimension.
- On Time-IMM, the event-sequence interface achieves lower value MAE and MSE than regular-grid interfaces for all five frozen TSFMs and in 8 of 9 dataset families.
- Against four trained irregular baselines in the same benchmark, frozen TSFMs using the event-sequence interface achieve lower next-event value and timing errors.
- On seven EasyTPP datasets, frozen TSFMs using the event-sequence interface provide competitive next-event timing forecasts relative to TPP baselines trained on each dataset.

2 Related Work

Our work connects three lines of research: zero-shot time series foundation models, task-trained methods for irregular time series, and event-sequence models for next-event timing. The first provides the frozen pretrained forecasters we study, the second defines the irregular forecasting problem and baselines, and the third motivates representing event timing with inter-event intervals while providing comparison context.

Time series foundation models. Recent general-purpose TSFMs target broad zero-shot forecasting through large-scale pretraining (Ansari et al., 2025; Gou et al., 2025; Das et al., 2024; Cohen et al., 2025; Khwaja et al., 2026). At inference time, a frozen TSFM typically takes a numeric context sequence $(\mathbf{s}_1, \ldots, \mathbf{s}_L)$ and returns forecasts for future sequence positions $(\hat{\mathbf{s}}_{L+1}, \ldots, \hat{\mathbf{s}}_{L+k})$. For regularly sampled time series, these positions are clock ticks; in this paper, the central question is what the positions should represent when observations are sparse and irregular. The experiments evaluate Chronos-2, CTSM 1.0, TimesFM 2.5, Toto 1.0, and Toto 2.0; section B.2 gives details about these TSFMs. Our work does not modify these architectures or fine-tune the TSFMs; it studies how irregular histories should be presented to frozen pretrained forecasters.

Domain-specific TSFMs such as MIRA, which is pretrained specifically for medical time series, fall outside our focus on general-purpose TSFMs (Li et al., 2025).

Irregular time-series modeling. Irregular time-series methods differ mainly in how they represent observation times before prediction. GRU-D uses observation masks and elapsed-time de-

cays so that recurrent states respond to informative missingness (Che et al., 2018). Latent ODE models represent irregular observations through continuous-time latent dynamics (Rubanova et al., 2019), while interpolation-prediction networks and multi-time attention networks build representations through learned interpolation or attention over observation times (Shukla & Marlin, 2019; 2021). Neural controlled differential equations similarly model an interpolated observation path with continuous-time dynamics (Kidger et al., 2020). These approaches are trained for the target task and encode irregularity through the architecture or preprocessing pipeline. Time-IMM is the closest benchmark for our problem because it focuses on irregular time-series forecasting (Chang et al., 2025). It spans multiple real-world domains and irregularity mechanisms, and it includes regular-grid adaptations and irregular time-series model baselines trained on each target dataset's training split. Our irregular time-series comparison uses four baseline methods from the Time-IMM benchmark suite: CRU, Latent ODE, Neural Flow, and tPatchGNN (Chang et al., 2025).

Temporal point processes. Temporal point process models represent continuous-time event sequences through arrival times, inter-event intervals, and, when available, event type labels. Many temporal point process models specify conditional intensities, while neural TPPs parameterize the history with recurrent states, continuous-time dynamics, attention, or Transformer architectures. Although TSFMs and neural TPPs can use similar sequence architectures, their outputs and training objectives differ: TSFMs are pretrained as numeric forecasters, whereas the TPP baselines learn event-time and event-type distributions; Table 3 compares their next-event timing points and event-type decisions. Intensity-free approaches instead model elapsed-time distributions directly, which motivates predicting event timing rather than deriving it from an intensity integral (Shchur et al., 2020). EasyTPP provides a complementary benchmark: it standardizes common event-sequence datasets, trained neural TPP baselines, and next-event time and event-type prediction tasks (Xue et al., 2024). Unlike Time-IMM series, EasyTPP sequences contain event times and a discrete type label for each event, not multivariate numerical values. They therefore test whether the interval coordinate also supports next-event timing on typed event sequences. Our EasyTPP comparison uses four baseline methods from that benchmark for next-event timing comparison: NHP, THP, AttNHP, and IFTPP.

Together, these lines of work clarify the gap addressed here. TSFMs provide powerful frozen sequence forecasters, but their native input axis is a clock-indexed grid. Irregular-time models and temporal point processes provide strong task-trained baselines for value and timing prediction, but they do not answer whether frozen TSFMs can be queried directly through an event-sequence representation. We study this interface directly: each observed event becomes a TSFM sequence position, its values and inter-event interval are the numeric inputs, and the forecast is evaluated on the next event's value and time. Detailed positioning is provided in section A.

3 METHODOLOGY

This section defines the input interface for applying a frozen TSFM to irregular time series. The interface specifies what each sequence position represents, what numeric dimensions are provided at that position, and which future positions are forecast. We first formulate irregular time-series forecasting as predicting future observed values and their timestamps. We then compare a regular-grid interface, which presents clock ticks including ticks without new observations, with an event-sequence interface, which presents observed events and exposes timing through an inter-event interval dimension.

3.1 IRREGULAR TIME-SERIES FORECASTING

An irregular time series consists of observations whose timestamps are not assumed to lie on a fixed sampling grid. Let $e_i = (t_i, \mathbf{x}_i)$ denote the i-th timestamped observation, where t_i is the timestamp and $\mathbf{x}_i \in \mathbb{R}^d$ is the observed value, with $t_1 < \cdots < t_i$. We allow $d = 1$ for scalar observations and $d > 1$ when multiple variables share timestamp t_i. The history before forecasting is the ordered tuple $\mathcal{H}_i = (e_1, \ldots, e_i)$.

With this numerical notation, the k-event irregular forecasting target is

$$\mathcal{Y}_{i,k} = \{(\mathbf{x}_{i+r}, t_{i+r})\}_{r=1}^{k}.$$

It contains the future observed values and the timestamps at which those values are observed. Given predictions $\{(\hat{\mathbf{x}}_{i+r}, \hat{t}_{i+r})\}_{r=1}^{k}$, value error is computed on $\hat{\mathbf{x}}_{i+r}$ and timing error is computed on $\hat{t}_{i+r}$. The horizon parameter k may be one for next-event forecasting or larger for multi-event irregular time-series forecasting. This formulation targets future observed events rather than values at prespecified query times, so both the future values and their timing are part of the target. The experiments below instantiate this setup with one-step prediction instances, so the target is the next observed value and its timestamp.

3.2 Regular-Grid Interfaces

A common way to adapt sparse observations to a regular-sequence forecaster is to choose a grid spacing h and convert the event history into values at grid times. Let $\mathcal{H}_i$ be the observed event history from section 3.1, and let $g_1, \ldots, g_{\ell_i}$ be the clock ticks that cover that history. We denote the regular-grid preprocessing rule by R_h. Given $\mathcal{H}_i$, this rule returns $\mathcal{G}_i = R_h(\mathcal{H}_i) = (\mathbf{v}_1, \ldots, \mathbf{v}_{\ell_i})$, where $\mathbf{v}_\ell$ is the value vector supplied to the TSFM at grid tick g_ℓ. The rule R_h also specifies how grid ticks without observations are represented, such as by imputed values or by an observation mask.

Regular-grid preprocessing introduces two coupled costs in the input representation. First, regular-grid preprocessing replaces the observed event history $\mathcal{H}_i$ with a constructed grid sequence $\mathcal{G}_i$. As a result, observed events are interleaved with imputed grid values, and periods with no observations still consume sequence positions. Second, the grid spacing h controls a tradeoff between timing fidelity and sequence length. A finer grid represents event times more precisely, but it uses more context positions and requires more future steps to reach the next event. A coarser grid shortens the sequence, but events that occur at different times within one grid cell can collapse into the same grid position. In the event-sequence interface, the observed events themselves are the input positions, and elapsed time is carried as an interval coordinate, so empty grid ticks are not inserted.

The grid also expands the forecast itself. With a regular-grid interface, the TSFM predicts every future clock tick up to the target event; the r-th future event appears roughly $m_{i,r} = \lceil (\sum_{q=1}^{r} \Delta t_{i+q})/h \rceil$ grid steps ahead, where $\Delta t_{i+q} = t_{i+q} - t_{i+q-1}$. In sparse series, many forecast positions therefore correspond to grid ticks with no event. Using a larger h reduces this count but blurs event time. The event-sequence interface avoids the dense path by making the next k observed events the next k sequence positions and forecasting their timing directly. Section B.3 gives an interface-level sequence-length summary for this comparison. The regular-grid interface therefore asks a frozen TSFM to model positions that are artifacts of the grid rather than observed events.

3.3 Event-Sequence TSFM Interface

Panels C–D of fig. 1 illustrate the event-sequence interface and forecast target. A frozen TSFM is queried through an ordered numeric context and returns forecasts over future sequence positions. The input interface determines what those positions represent: regular-grid interfaces supply clock ticks, whereas the event-sequence interface supplies observed events. Therefore, the next TSFM continuation position corresponds to the next observed event. For each observed event, we create one input element $\mathbf{u}_i = [\mathbf{x}_i, \phi(\Delta t_i)] \in \mathbb{R}^{d+1}$, where $\Delta t_i = t_i - t_{i-1}$ for $i > 1$, $\Delta t_1 = 0$, and ϕ is an interval transform. The TSFM receives the event-sequence object $\mathcal{E}_i = (\mathbf{u}_1, \ldots, \mathbf{u}_i)$ and forecasts future event-order elements $\widehat{\mathcal{U}}_{i,k} = (\hat{\mathbf{u}}_{i+1}, \ldots, \hat{\mathbf{u}}_{i+k})$. At each future event step, the predicted element contains value dimensions $\hat{\mathbf{x}}_{i+r}$ and an interval dimension. The interval dimension gives $\Delta\hat{t}_{i+r}$ after applying ϕ^{-1}. This changes the continuation target from future clock ticks to future observed events while preserving event-time information through decoded inter-event intervals. Section B.1 shows that this event-indexed sequence, together with the initial timestamp, preserves the original values, event order, and event times.

Figure 2 illustrates the diversity of log-transformed Time-IMM interval histories, ranging from the strongly repeated FNSPID pattern to less visually regular examples. A shuffle-based autocorrelation check finds stronger short-lag autocorrelation than 90% of shuffled versions of the same sequence for 54 of 66 nonconstant histories (section B.5). This order dependence suggests that past intervals can help predict future intervals, providing temporal structure that TSFMs can exploit.

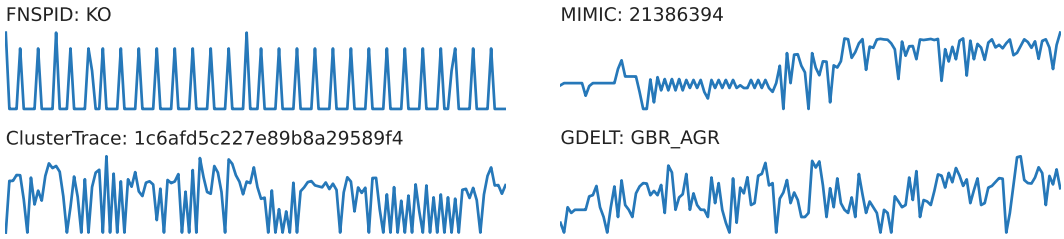


Figure 2: Examples of log-transformed inter-event interval sequences from Time-IMM.

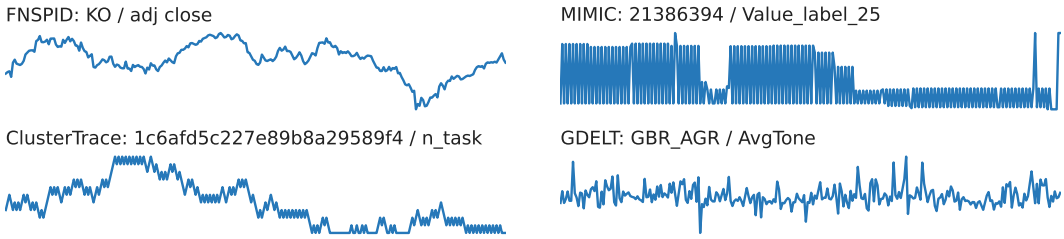


Figure 3: Examples of event-indexed value sequences from Time-IMM.

Figure 3 gives a companion view of value channels from the same four Time-IMM series. Section B.5 also applies the same shuffle-based autocorrelation check independently to value channels: 383 of 497 eligible channels have stronger short-lag autocorrelation than 90% of shuffled versions of the same channel. Together with the examples in fig. 3, this shows that event-indexed value histories can also contain local regimes, trends, or repeated patterns rather than independent draws from a marginal value distribution. This motivates including event timing in the event-sequence representation; the experiments evaluate both next-event value and timing from this representation.

We evaluate two interval encodings for ϕ. Let τ_0 denote one unit of the dataset's stored time scale. The raw interval uses $\phi(\Delta t) = \Delta t$ and preserves the physical time unit. The log interval uses $\phi(\Delta t) = \log(1 + \Delta t/\tau_0)$; it preserves interval order while compressing large gaps. TSFMs differ in input normalization, numeric-range sensitivity, multivariate coupling, and pretraining distribution. We therefore evaluate both interval encodings rather than assuming one is universally preferable; section B.2 gives model-specific input-format details.

This interface requires no architecture change, fine-tuning, or continuous-time inference routine; it only changes the ordered numeric sequence supplied at inference time.

4 Experiments

This section evaluates the event-sequence interface for next-event value and timing forecasts. On Time-IMM, we compare event-sequence and regular-grid interfaces for the same frozen TSFMs; we also compare the event-sequence forecasts with trained irregular baselines that either jointly predict timing or receive oracle time. EasyTPP provides comparison to trained temporal point-process baselines. We report the aggregate comparisons needed for these evaluations; section E uses Time-IMM dataset characteristics to explain performance gaps in comparisons with regular-grid interfaces and trained irregular baselines; section D collects supporting analyses and sensitivity results, and section F gives complete result tables.

4.1 Time-IMM Benchmark and Setup

Time-IMM is the primary benchmark because it provides irregular time-series forecasting problems across nine dataset families and includes irregular time-series model baselines (Chang et al., 2025). We evaluate 85 processed Time-IMM series from CESNET, ClusterTrace, EPA-Air, FNSPID, GDELT, ILINet, MIMIC, RepoHealth, and StudentLife. We evaluate five TSFMs: Chronos-2,

CTSM 1.0, TimesFM 2.5, Toto 1.0, and Toto 2.0; sections B.2 and D.4 gives the evaluated checkpoints, input formats, and descriptive cross-TSFM results. All Time-IMM comparisons use the same next-event task: given the history through event i, every method is scored on the value of event $i + 1$, and methods that forecast timing are also scored by timing error for event $i + 1$. Following the released Time-IMM evaluation pipeline, Time-IMM value errors are reported on the normalized value scale. We also report relative timing errors because elapsed times between events vary widely across dataset families. Section C.1 gives the exact evaluation setup: how each next-event prediction is formed, what each method predicts, and which method groups are compared; section C.2 gives the metric definitions.

4.2 Event-Sequence Versus Regular-Grid TSFM Interfaces

The interface comparison asks whether the event-sequence interface improves value forecasting by frozen TSFMs on irregular time series relative to regular-grid interfaces. For each TSFM, we evaluate a small set of configurations for each interface. Event-sequence configurations differ only in interval encoding: raw or log-transformed inter-event intervals. Regular-grid configurations combine a regular-grid variant with an output budget. The regular-grid variants are sample-and-hold, linear interpolation, and zero fill. When the TSFM input format supports mask dimensions, we also evaluate sample-and-hold with observation masks. The output budget is 64, 128, or 256 future grid steps. Section C.3 gives the regular-grid construction details. For each dataset family and TSFM, validation normalized value MAE selects the reported configuration for each interface. For the event-sequence interface, it selects the interval encoding. For the regular-grid interface, it selects both the regular-grid variant and the output budget.

This interface comparison reports value error only. Under a regular-grid interface, a TSFM forecasts values at prespecified future grid ticks; it does not predict whether an event occurs at a tick. For scoring, the true next-event timestamp is used to choose the readout grid tick: we take the first future grid tick at or after that timestamp and compare its value forecast with the ground-truth next-event value. Because regular-grid variants do not predict that timestamp, they are not scored on timing.

Table 1: Time-IMM interface comparison. Lower MAE/MSE is better.

TSFM	Interface	Selected Configs*	MAE	MSE
Chronos-2	Event sequence	log=6, raw=3	0.363	0.473
	Regular grid	hold-64=3, mask-256=1, mask-64=5	0.402	0.583
CTSM 1.0	Event sequence	raw=9	0.370	0.470
	Regular grid	hold-128=1, hold-256=3, hold-64=2, linear-64=2, zero-64=1	0.395	0.547
TimesFM 2.5	Event sequence	raw=9	0.363	0.478
	Regular grid	hold-256=1, hold-64=6, zero-64=2	0.408	0.604
Toto 1.0	Event sequence	log=5, raw=4	0.368	0.504
	Regular grid	hold-128=1, hold-64=5, zero-64=3	0.437	0.703
Toto 2.0	Event sequence	log=5, raw=4	0.353	0.484
	Regular grid	hold-128=1, hold-64=3, linear-64=2, mask-64=2, zero-64=1	0.420	0.664

* Entries use configuration=count, where count is the number of dataset families selected by validation MAE; e.g., `hold-64=3` means sample-and-hold with 64 output steps was selected for three families.

Table 1 compares event-sequence and regular-grid interfaces for each TSFM on normalized value MAE and MSE, averaged over the nine Time-IMM dataset families. In every TSFM block, the event-sequence interface has lower MAE/MSE than the regular-grid interface. For example, Chronos-2 is 0.363/0.473 versus 0.402/0.583, and Toto 2.0 is 0.353/0.484 versus 0.420/0.664. The same pattern holds for CTSM 1.0, TimesFM 2.5, and Toto 1.0. Section D.1 reports the same comparison for 64-, 128-, and 256-step regular-grid output budgets. At the dataset-family level, the event-sequence interface outperforms regular-grid interfaces on value MAE and value MSE in 8 of 9 Time-IMM families. The mean family gap is 0.0488 for value MAE and 0.1384 for value MSE, where positive gaps mean the regular-grid interface performs worse. Section D.2 gives the corresponding family-level paired gaps. Sections D.3 and D.4 report interval-encoding and cross-TSFM sensitivity results. Together, these results show that the event-sequence interface achieves lower value MAE and MSE than regular-grid interfaces across all five frozen TSFMs and in 8 of 9 dataset families, supporting the interpretation that the input interface can be an important performance bottleneck for frozen

TSFMs on irregular time series. Section E examines how grid readout length and one-step value changes relate to the observed family-level gaps.

4.3 EVENT-SEQUENCE TSFM INTERFACE VERSUS TRAINED IRREGULAR BASELINES

This comparison evaluates frozen TSFMs using the event-sequence interface against four trained irregular-model adaptations: CRU, Latent ODE, Neural Flow, and tPatchGNN; section C.4 gives their implementation details and selection rationale.

In the joint timing prediction setup, we attach a learned scalar interval-prediction head to each trained irregular model. The resulting model predicts both the next value and the elapsed time until the next event, so we report both value and timing errors. In the oracle time setup,[2] each trained irregular model receives the ground-truth next-event timestamp as the query time. Because the timestamp is given, this setup evaluates only the value forecast at that timestamp and does not evaluate timing prediction. Validation normalized value MAE selects the raw/log interval encoding for each TSFM using the event-sequence interface. For the joint timing prediction setup, validation normalized value MAE also selects whether the interval-prediction head is trained with raw or log-transformed interval targets. The oracle time setup has no raw/log interval-target choice because the next-event timestamp is supplied rather than predicted.

Table 2: Time-IMM comparison of the event-sequence TSFM interface with trained irregular baselines. Metrics are averaged over dataset families; timing errors use the same configurations selected by validation value MAE. For all error columns, lower is better.*

Model	Setup	Selected Configs	Value Error		Timing Error	
			MAE	MSE	MAE	MSE
Chronos-2	Event sequence	log=6, raw=3	0.363	0.473	1.377	115.7
CTSM 1.0	Event sequence	raw=9	0.370	**0.470**	1.647	114.1
TimesFM 2.5	Event sequence	raw=9	0.363	0.478	1.410	**112.8**
Toto 1.0	Event sequence	log=5, raw=4	0.368	0.504	1.411	116.3
Toto 2.0	Event sequence	log=5, raw=4	**0.353**	0.484	**1.334**	112.9
CRU	Joint timing prediction	log=6, raw=3	1.043	1.876	**1.988**	**128.1**
CRU	Oracle time	oracle=9	0.671	0.975	–	–
Latent ODE	Joint timing prediction	log=7, raw=2	0.741	1.124	2.187	128.7
Latent ODE	Oracle time	oracle=9	0.673	0.985	–	–
Neural Flow	Joint timing prediction	log=8, raw=1	0.678	0.969	2.178	129.2
Neural Flow	Oracle time	oracle=9	0.676	0.987	–	–
tPatchGNN	Joint timing prediction	log=8, raw=1	**0.560**	**0.704**	2.179	129.3
tPatchGNN	Oracle time	oracle=9	**0.606**	**0.800**	–	–

* Value errors are standard-deviation normalized and timing errors are relative to the series timing scale; both are dimensionless. Bold entries are metric-specific minima within each setup group.

Table 2 reports normalized value error and relative timing error, averaged over the nine Time-IMM dataset families. For value MAE, the best frozen TSFM using the event-sequence interface achieves 0.353, compared with 0.560 for the best joint-timing baseline and 0.606 for the best oracle-time baseline. For timing MAE, the best frozen TSFM using the event-sequence interface achieves 1.334, compared with 1.988 for the best joint-timing baseline. At the dataset-family level, frozen TSFMs using the event-sequence interface outperform these baselines on value MAE in 9 of 9 families and value MSE in 8 of 9 families under both setups, and on timing MAE and MSE in 9 of 9 families under joint timing prediction. Section D.2 reports the corresponding family-level paired gaps. Section D.5 reports the Time-IMM interval-history perturbation controls for next-event timing. Section E analyzes how value-change and value-predictability diagnostics help explain the family-level performance gaps. Taken together, these results show that frozen TSFMs using the event-sequence interface achieve lower next-event value MAE and MSE than the four trained irregular baselines under both joint timing prediction and oracle time. Notably, the oracle-time baselines receive the ground-truth next-event timestamp yet still have worse value MAE and MSE. In the joint timing prediction setup, frozen TSFMs using the event-sequence interface also achieve lower timing MAE and MSE than the trained irregular baselines.

[2]The term denotes access to ground-truth timing, not a performance upper bound.

4.4 EASYTPP TIMING AND EVENT-TYPE RESULTS

We use EasyTPP to evaluate whether the timing component of the event-sequence interface transfers to event sequences with discrete event types (Xue et al., 2024). In this benchmark, models are evaluated on next-event prediction, with time prediction and event-type prediction reported separately. The benchmark provides 383,397 timing prediction instances from seven datasets. Following the released EasyTPP evaluation pipeline, EasyTPP timing errors are reported directly in each dataset's native time unit.

The TPP baselines are NHP, THP, AttNHP, and IFTPP; section C.5 gives the model descriptions and selection rationale. For EasyTPP TSFM configurations, event type is encoded with either one-hot encoding or frequency-ordered scalar encoding; section B.4 defines these encodings, and section D.6 compares them on EasyTPP. For each dataset, validation timing MAE selects the TSFM, raw/log interval encoding, and event-type representation. In the TSFM selection column, R denotes raw intervals, L denotes log-transformed intervals, and OH denotes one-hot event-type encoding. For the TPP results in table 3, each dataset uses the trained TPP baseline with the lowest test timing MAE. The table reports that same baseline's timing MSE and event-type error.

Table 3: EasyTPP timing and event-type comparison. Within each TSFM/TPP metric pair, bold marks the smaller unrounded error; lower error is better.[†]

Dataset	TSFM Sel.	TPP	Timing MAE		Timing MSE		Type Err.	
			TSFM	TPP	TSFM	TPP	TSFM	TPP
Amazon	TimesFM 2.5 (R/OH)	IFTPP	**0.262**	0.300	**0.149**	0.210	0.713	**0.644**
Earthquake	TimesFM 2.5 (L/OH)	AttNHP	**0.761**	0.885	**1.854**	2.490	0.625	**0.537**
Retweet	TimesFM 2.5 (L/OH)	IFTPP	2326	**2289**	2.49×10^8	2.47×10^8	0.406	**0.399**
StackOverflow	TimesFM 2.5 (L/OH)	AttNHP	**0.687**	0.785	**1.279**	1.714	**0.603**	0.626
Taobao	TimesFM 2.5 (L/OH)	IFTPP	0.045	**0.042**	0.017	**0.017**	0.396	**0.387**
Taxi	TimesFM 2.5 (L/OH)	IFTPP	**0.162**	0.205	**0.099**	0.159	0.212	**0.082**
Volcano	CTSM 1.0 (L/OH)	IFTPP	953.1	**886.5**	1.70×10^7	1.50×10^7	–*	–*

[†] Timing MAE is in each dataset's native time unit; timing MSE is in squared native units. Type Err. is the fraction of evaluated next-event predictions with an incorrect event type.
* Type error is not applicable to Volcano because it contains only one event type.

Table 3 shows that TSFMs have lower timing MAE and MSE on 4 of 7 datasets. These results show that frozen TSFMs using the event-sequence interface can predict when the next event occurs on TPP-style event sequences. Excluding Volcano, which contains only one event type, TSFMs have lower event-type error on 1 of the remaining 6 datasets. The evaluated TSFMs encode categorical event types as numeric dimensions and map each forecast back to an event label, whereas TPP baselines predict native event-type probabilities; this output-interface difference may contribute to the TPP baselines' lower event-type error. Overall, frozen TSFMs using the event-sequence interface are competitive with TPP baselines trained on each dataset for next-event timing, while the trained TPP baselines remain stronger for event-type prediction. Section D.5 reports the EasyTPP interval-history perturbation controls for next-event timing.

5 CONCLUSION

This paper studies an event-sequence interface that presents sparse irregular histories to frozen TSFMs as sequences of observed events and inter-event intervals. Across all five evaluated TSFMs, the event-sequence interface achieves lower value MAE and MSE than regular-grid interfaces on Time-IMM. Frozen TSFMs using the event-sequence interface also achieve lower next-event value and timing errors than four trained irregular baselines on Time-IMM and provide competitive next-event timing forecasts relative to TPP baselines trained on each EasyTPP dataset. Together, these results support the paper's central claim: time series foundation models can be effective forecasters for irregular time series when histories are represented as event sequences.

AI USE STATEMENT

We used generative AI tools at several stages of this work. During research development, they supported brainstorming and provided feedback on the methodology and experimental design. Dur-

ing implementation, they assisted with editing experimental code. During manuscript preparation, they helped improve the clarity and readability of the text. The authors critically evaluated all AI-generated suggestions, reviewed and tested all AI-assisted code changes, and independently verified the experimental results. All methodological decisions, interpretations, and conclusions were made by the authors.

REFERENCES

Abdul Fatir Ansari, Oleksandr Shchur, Jaris Küken, Andreas Auer, Boran Han, Pedro Mercado, Syama Sundar Rangapuram, Huibin Shen, Lorenzo Stella, Xiyuan Zhang, et al. Chronos-2: From univariate to universal forecasting. *arXiv preprint arXiv:2510.15821*, 2025.

Marin Biloš, Johanna Sommer, Syama Sundar Rangapuram, Tim Januschowski, and Stephan Günnemann. Neural flows: Efficient alternative to neural odes. *Advances in neural information processing systems*, 34:21325–21337, 2021.

Patricio Cerda and Gaël Varoquaux. Encoding high-cardinality string categorical variables. *IEEE Transactions on Knowledge and Data Engineering*, 34(3):1164–1176, 2020.

Ching Chang, Jeehyun Hwang, Yidan Shi, Haixin Wang, Wei Wang, Wen-Chih Peng, and Tien-Fu Chen. Time-IMM: A dataset and benchmark for irregular multimodal multivariate time series. *Advances in Neural Information Processing Systems*, 38, 2025.

Zhengping Che, Sanjay Purushotham, Kyunghyun Cho, David Sontag, and Yan Liu. Recurrent neural networks for multivariate time series with missing values. *Scientific reports*, 8(1):6085, 2018.

Ben Cohen, Emaad Khwaja, Youssef Doubli, Salahidine Lemaachi, Chris Lettieri, Charles Masson, Hugo Miccinilli, Elise Ramé, Qiqi Ren, Afshin Rostamizadeh, et al. This time is different: An observability perspective on time series foundation models. *Advances in neural information processing systems*, 38:50907–50951, 2025.

Abhimanyu Das, Weihao Kong, Rajat Sen, and Yichen Zhou. A decoder-only foundation model for time-series forecasting. In *International Conference on Machine Learning*, pp. 10148–10167. PMLR, 2024.

Google Research. TimesFM 2.5. `https://huggingface.co/google/timesfm-2.5-200m-pytorch`, 2025. Model card.

Liang Gou, Archit Khare, Praneet Pabolu, Prachi Patel, Joseph Ross, Hercy Shen, Jingze Sun, Kristal Curtis, Vedant Dharnidharka, Abhinav Mathur, et al. Cisco time series model technical report. *arXiv preprint arXiv:2511.19841*, 2025.

Rob J Hyndman and George Athanasopoulos. *Forecasting: principles and practice*. OTexts, 2018.

Emaad Khwaja, Chris Lettieri, Gerald Woo, Eden Belouadah, Marc Cenac, Guillaume Jarry, Enguerrand Paquin, Xunyi Zhao, Viktoriya Zhukov, Othmane Abou-Amal, et al. Toto 2.0: Time series forecasting enters the scaling era. *arXiv preprint arXiv:2605.20119*, 2026.

Patrick Kidger, James Morrill, James Foster, and Terry Lyons. Neural controlled differential equations for irregular time series. *Advances in neural information processing systems*, 33:6696–6707, 2020.

Hao Li, Bowen Deng, Chang Xu, Zhiyuan Feng, Viktor Schlegel, Yu-Hao Huang, Yizheng Sun, Jingyuan Sun, Kailai Yang, Yiyao Yu, et al. MIRA: Medical time series foundation model for real-world health data. *Advances in neural information processing systems*, 38:98657–98685, 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.

Hongyuan Mei, Chenghao Yang, and Jason Eisner. Transformer embeddings of irregularly spaced events and their participants. In *International conference on learning representations*, 2022.

Yulia Rubanova, Ricky TQ Chen, and David K Duvenaud. Latent ordinary differential equations for irregularly-sampled time series. *Advances in neural information processing systems*, 32, 2019.

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. URL https://openreview.net/forum?id=HygOjhEYDH.

Satya Narayan Shukla and Benjamin M Marlin. Interpolation-prediction networks for irregularly sampled time series. *arXiv preprint arXiv:1909.07782*, 2019.

Satya Narayan Shukla and Benjamin M Marlin. A survey on principles, models and methods for learning from irregularly sampled time series. *arXiv preprint arXiv:2012.00168*, 2020.

Satya Narayan Shukla and Benjamin M Marlin. Multi-time attention networks for irregularly sampled time series. *arXiv preprint arXiv:2101.10318*, 2021.

Siqiao Xue, Xiaoming Shi, Zhixuan Chu, Yan Wang, Hongyan Hao, Fan Zhou, Caigao Jiang, Chen Pan, James Y. Zhang, Qingsong Wen, Jun Zhou, and Hongyuan Mei. EasyTPP: Towards open benchmarking temporal point processes. In *International conference on learning representations*, 2024.

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.

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 Additional Related-Work Positioning

- **TSFM forecasting on regular grids.** This line provides powerful pretrained sequence forecasters on clock-indexed data. This paper reindexes sparse irregular observations by observed events so the next TSFM position is the next event, avoiding regular-grid fill artifacts. The contribution is an input-interface change, not new TSFM training or architecture.
- **Imputation and interpolation pipelines.** These pipelines convert irregular samples to a regular grid before forecasting. This paper shows that frozen TSFMs can be used without first choosing a clock grid or fill rule, and empirically compares against regular-grid interfaces.
- **Irregular-time neural models.** These models learn dynamics from irregular timestamps using task-specific training. This paper tests whether off-the-shelf frozen TSFMs can be competitive through an input-interface change alone. It does not claim learned continuous-time dynamics.
- **Native TPPs and marked processes.** These models define event-time and mark likelihoods with process objectives. Instead, this paper uses inter-event intervals as numeric input/output coordinates for frozen TSFMs and evaluates next-event point forecasts for values and timing, together with decoded event-type predictions. It does not claim likelihoods, survival terms, calibrated event-time laws, or native mark distributions for TSFMs.

B Interface, TSFM Inputs, and Representations

This section describes the event-sequence representation, the evaluated TSFM input formats, event-type encodings for EasyTPP, and the interval-predictability analysis used to motivate timing prediction.

B.1 Event-Order Representation and Point Readout

The event-sequence interface represents an irregular history as an ordered numeric sequence indexed by observed events. The following lemma states what information this representation preserves.

Definitions and assumptions. Treat an event history as an ordered tuple $\mathcal{H}_i = ((t_1, \mathbf{x}_1), \ldots, (t_i, \mathbf{x}_i))$. Assume the event times are strictly ordered, $t_1 < \cdots < t_i$. Define $\Delta t_1 = 0$. For $j = 2, \ldots, i$, define the inter-event interval $\Delta t_j = t_j - t_{j-1}$. Inter-event intervals satisfy $\Delta t_j > 0$ for $j = 2, \ldots, i$, while the first padded interval is nonnegative by construction. The interval transform ϕ is injective on the nonnegative interval domain and has an inverse on its image. Let τ_0 denote one unit of the dataset's stored time scale. The evaluated encodings are special cases: the raw-interval encoding uses $\phi(s) = s$, and the log-interval encoding uses $\phi(s) = \log(1 + s/\tau_0)$. Both are injective for $s \geq 0$, with inverse decoders $\phi^{-1}(r) = r$ and $\phi^{-1}(r) = \tau_0\big(\exp(r) - 1\big)$, respectively. The initial timestamp is retained externally or the sequence is recentered so that $t_1 = 0$.

Lemma. Under the assumptions above, the event-indexed representation

$$\mathcal{R}(\mathcal{H}_i) = ((\mathbf{x}_1, \phi(0)), (\mathbf{x}_2, \phi(\Delta t_2)), \ldots, (\mathbf{x}_i, \phi(\Delta t_i)))$$

together with the retained or recentered initial timestamp uniquely determines the ordered history $\mathcal{H}_i$. Thus, reindexing the sequence by observed-event order preserves the observed values, event order, and event times. If a predicted next element contains a value forecast $\hat{\mathbf{x}}_{i+1}$ and a decoded nonnegative interval $\Delta\hat{t}_{i+1}$, it defines the one-step point forecast $(\hat{\mathbf{x}}_{i+1}, \hat{t}_{i+1})$ with $\hat{t}_{i+1} = t_i + \Delta\hat{t}_{i+1}$.

Proof sketch. Values and event order are retained directly in the ordered sequence. Since ϕ is injective on the interval domain, each transformed interval for $j = 2, \ldots, i$ uniquely determines Δt_j, and the first coordinate encodes the padded value $\Delta t_1 = 0$. Given the retained or recentered initial timestamp, arrival times are recovered by accumulation:

$$t_j = t_1 + \sum_{q=2}^{j} \Delta t_q.$$

Therefore replacing clock index with event index changes the sequence axis but does not discard event-time information. This establishes the formal bridge used by the method: a frozen TSFM continuation position can be aligned with the next observed event, and the resulting next sequence element can encode both value and timing. The forecasting experiments evaluate this interface on the next-event task.

B.2 Interval Encodings, Evaluated TSFMs, and Input Formats

Both interval encodings used in section 3.3 are decoded before computing timing metrics. For raw intervals, $\phi^{-1}(z) = z$. For log-transformed intervals, $\phi^{-1}(z) = \tau_0\big(\exp(z) - 1\big)$. The evaluation therefore compares raw and log-transformed interval encodings in the original time unit of each dataset. The evaluated TSFMs are Chronos-2, CTSM 1.0, TimesFM 2.5, Toto 1.0, and Toto 2.0 (Ansari et al., 2025; Gou et al., 2025; Das et al., 2024; Cohen et al., 2025; Khwaja et al., 2026).

- **Chronos-2.** We use the `amazon/chronos-2` checkpoint. Chronos-2 is an encoder-only TSFM for univariate, multivariate, and covariate-informed zero-shot forecasting (Ansari et al., 2025). Its group attention mechanism shares information across related series, variates, targets, and covariates without treating all variables as one long time axis. The `amazon/chronos-2` checkpoint is trained on subsets of the Chronos Datasets and GIFT-Eval Pretrain plus synthetic univariate and multivariate data; in the technical report, the multivariate and covariate-informed training tasks are generated synthetically by imposing dependencies on univariate generators. In our evaluation, Chronos-2 is used with the event value dimensions and interval dimension represented together as one multivariate time series.
- **CTSM 1.0.** We use the `cisco-ai/cisco-time-series-model-1.0` checkpoint. CTSM 1.0 is a univariate zero-shot forecaster built around decoder-only transformer layers

with TimesFM-style patch forecasting (Gou et al., 2025). Its main architectural feature is a multiresolution context: coarse- and fine-resolution histories are aligned at the forecast origin so long historical context can inform fine-resolution forecasts. The released 1.0 checkpoint is trained from scratch rather than continued from TimesFM weights, with a training mixture dominated by internal Splunk observability metrics plus public and synthetic series. In our evaluation, each value dimension and the interval sequence are represented as separate univariate time series.

- **TimesFM 2.5.** We use the `google/timesfm-2.5-200m-pytorch` checkpoint (Google Research, 2025). TimesFM is a decoder-only TSFM trained for zero-shot forecasting on large real and synthetic time-series corpora (Das et al., 2024). It tokenizes univariate histories into contiguous input patches and predicts future output patches, reducing the number of autoregressive decoding steps relative to point-by-point generation. The 2.5 checkpoint is pretrained on GIFT-Eval Pretrain, Wikimedia Pageviews, Google Trends top queries, and synthetic or augmented data, while retaining the TimesFM patch objective that predicts future output patches from earlier input patches. In our evaluation, each value dimension and the interval sequence are represented as separate univariate time series.
- **Toto 1.0.** We use the `Datadog/Toto-Open-Base-1.0` checkpoint. Toto 1.0 is a decoder-only TSFM designed for multivariate observability time series (Cohen et al., 2025). It combines patch embeddings with factorized time- and variate-attention layers, causal per-variate scaling, and a Student-t mixture output head for probabilistic forecasts. Toto 1.0 is trained on an observability-heavy corpus of roughly 2.36 trillion time-series points, including public datasets and synthetic data, with a next-patch probabilistic objective and composite robust loss. In our evaluation, Toto 1.0 is used with the event value dimensions and interval dimension represented together as one multivariate time series.
- **Toto 2.0.** We use the `Datadog/Toto-2.0-313m` checkpoint. Toto 2.0 keeps the decoder-only patched variate-time transformer backbone and changes the forecasting recipe for scalable parallel decoding (Khwaja et al., 2026). Its contiguous patch masking objective supports single-pass prediction of masked future patches, and its quantile output head replaces the Student-t mixture head used in Toto 1.0. The evaluated 313M checkpoint is part of a Toto 2.0 family trained with a shared u-μP scaling recipe on Datadog observability metrics and synthetic data, using pinball loss and NorMuon optimization. In our evaluation, Toto 2.0 is used with the event value dimensions and interval dimension represented together as one multivariate time series.

The evaluated TSFMs fall into two input formats in our experiments.

- **Multivariate input: Chronos-2, Toto 1.0, and Toto 2.0.** The value dimensions and interval dimension are represented together as a multivariate time series. The next-value and interval forecasts can therefore condition on both value and interval histories in the same model call. Changing from raw to log-transformed intervals can affect both value and timing forecasts.
- **Univariate input: CTSM 1.0 and TimesFM 2.5.** Each value dimension and the interval sequence are represented independently as separate univariate time series. Value forecasts therefore use event-indexed value histories, while the interval forecast uses the event-indexed interval history. Changing from raw to log-transformed intervals affects only the interval forecast.

Across both cases, the common interface gain is that the next TSFM position corresponds to the next observed event rather than to a synthetic grid tick. All evaluated TSFMs use a 1024-position context budget; for CTSM 1.0, this budget is split between 512 coarse-context positions and 512 fine-context positions. Regular-grid variants use the same 1024-position budget after grid preprocessing. Table 4 summarizes these evaluated input formats and forecast families.

B.3 Sequence Length Under Regular-Grid and Event-Sequence Interfaces

The sequence-length tradeoff in section 3.2 can be summarized by comparing the number of input and output positions used by each interface. Let i be the number of observed history events and k the number of future observed events to forecast. Let η denote the average number of grid ticks per

Table 4: Evaluated TSFM input formats and model-side forecast families.

TSFM	Evaluated input format	Forecast family
Chronos-2	Multivariate numeric sequence	Mean and quantile forecasts
CTSM 1.0	Separate univariate numeric series	Mean and quantile forecasts
TimesFM 2.5	Separate univariate numeric series	Mean and quantile forecasts
Toto 1.0	Multivariate numeric sequence	Student-t mixture forecasts
Toto 2.0	Multivariate numeric sequence	Quantile forecasts

observed event over the history-plus-forecast span at grid width h. Then the regular-grid interface uses $O(\eta i)$ input positions and $O(\eta k)$ output positions, while the event-sequence interface uses $O(i)$ input positions and $O(k)$ output positions.

Table 5: Sequence-position scaling for a sparse history of i observed events and a forecast of k future observed events.

Interface	Input sequence	Output sequence
Regular grid	$O(\eta i)$	$O(\eta k)$
Event sequence	$O(i)$	$O(k)$

All five evaluated TSFMs use dense attention along the temporal patch axis, so the attention-score computation is quadratic in the number of input patches. With fixed patch size, the corresponding history-length factor changes from $O(\eta^2 i^2)$ under a regular grid to $O(i^2)$ under the event-sequence interface.

B.4 Event-Type Representations

The main event-sequence interface treats value and timing as numeric dimensions. The EasyTPP timing experiments additionally require a numeric representation of the discrete event type, because the evaluated TSFMs do not provide a native categorical output head. Event type must therefore be represented numerically before it can be passed to a TSFM.

For a dataset with c event types, the original event IDs are categorical event labels rather than numeric measurements. A one-hot representation respects this categorical structure, but it creates c sparse dimensions; this is the standard tradeoff in categorical encoding, especially when cardinality grows (Cerda & Varoquaux, 2020). We therefore evaluate two encodings for exposing event type to a zero-shot numeric forecaster.

- **One-hot encoding.** Each event type is represented by c binary dimensions, and the predicted type is decoded by selecting the dimension with the largest forecasted value. This respects the categorical structure of event labels, but it increases the input dimensionality and turns categorical prediction into a numeric dimension-selection problem.
- **Frequency-ordered scalar encoding.** We count event types in the training split and order them from most to least frequent, assigning rank 0 to the most frequent type, rank 1 to the next most frequent type, and so on. After the TSFM produces a scalar point forecast, the forecast is rounded and clipped to the nearest valid rank and mapped back to the original event type. The motivation is the rounded decoding step: frequency ordering makes the scalar regions around valid ranks prevalence-aware, because common event types occupy compact low-rank values. Rare event types remain reachable, but the forecast must move toward their assigned ranks when the recent history supports them.

In both encodings, the interval dimension remains the first dimension and is represented with either the raw inter-event interval or its log transform. The event-type vocabulary and any frequency ordering are estimated from the training split and then held fixed for evaluation.

B.5 Interval and Value Order Dependence

We use a shuffle-based autocorrelation check to measure short-lag order dependence in the inter-event interval sequences from Time-IMM. Autocorrelation is a standard measure of lagged depen-

dence in a time series (Hyndman & Athanasopoulos, 2018). If an interval sequence has stronger short-lag autocorrelation than shuffled versions of the same values, its temporal order contains lagged structure beyond its marginal interval distribution.

For each Time-IMM series, we transform the inter-event interval sequence as $y_i = \log(1 + \Delta t_i/\tau_0)$. We only keep nonconstant transformed series because autocorrelation is undefined for constant sequences. This leaves 66 interval sequences. All 66 have at least $3 \times 32 = 96$ finite intervals, so the fixed lag range $1, \ldots, 32$ stays within the first third of every sequence while keeping the same lag range across series. For each series, we compute sample autocorrelations for lags $\tau = 1, \ldots, 32$ and summarize the series by

$$a = \max_{1 \le \tau \le 32} |\hat{\rho}(\tau)|.$$

We then randomly permute that series' interval values 1000 times and recompute a for each shuffled sequence. This preserves each series' interval values and length while removing temporal order. For every shuffled sequence, we again take the maximum over lags $1, \ldots, 32$, so the shuffled percentile is a threshold for the same max-over-lags statistic rather than for one fixed lag. For a percentile $p \in \{90, 95, 99\}$, we use the p-th percentile of those 1000 shuffled statistics as the comparison threshold. We count a series as exceeding that threshold when its observed statistic a is larger than the corresponding shuffled percentile. Such a series has stronger short-lag dependence than most shuffled versions of the same interval sequence. Across the 66 nonconstant series, the threshold summary is:

- Shuffled 90th percentile: 54/66 series (82%) exceed this threshold.
- Shuffled 95th percentile: 52/66 series (79%) exceed this threshold.
- Shuffled 99th percentile: 50/66 series (76%) exceed this threshold.

Thus, most nonconstant interval series contain short-lag order structure beyond what is explained by their marginal interval distribution alone. This order dependence suggests that past intervals can be useful for forecasting future intervals.

We apply the same shuffle-based autocorrelation check to event-indexed value sequences, treating each value channel independently. For each Time-IMM series and value channel, we keep finite nonconstant channels with at least $3 \times 32 = 96$ observations, compute the same statistic $a = \max_{1 \le \tau \le 32} |\hat{\rho}(\tau)|$, and compare it with 1000 shuffled versions of that channel. This leaves 497 eligible value channels across 85 Time-IMM series. The threshold summary is:

- Shuffled 90th percentile: 383/497 channels (77%) exceed this threshold.
- Shuffled 95th percentile: 372/497 channels (75%) exceed this threshold.
- Shuffled 99th percentile: 350/497 channels (70%) exceed this threshold.

Thus, most eligible value channels also contain short-lag order structure beyond what is explained by their marginal value distribution alone. The event-sequence representation therefore exposes the ordered structure in both value and inter-event interval sequences to the forecasting model.

C Evaluation Setup, Metrics, and Baselines

This appendix section collects the experimental definitions that support the main Time-IMM and EasyTPP results. It specifies how next-event predictions are formed, which method groups are compared, metric normalizations, regular-grid variants, and baseline choices behind the aggregate comparisons. The main experiment section reports the key findings; the details here make the comparisons reproducible without interrupting the main narrative.

C.1 Evaluation Setup and Compared Method Groups

This subsection explains how each next-event prediction is formed, what each method is asked to predict, and which method groups are compared.

Time-IMM instances. In the processed Time-IMM evaluation, each event has a shared timestamp and a fixed value vector $\mathbf{x}_i \in \mathbb{R}^d$ over the processed value dimensions. Dataset-level masks determine which value dimensions are evaluated; masked dimensions are excluded from the value metric. All Time-IMM comparisons use the same observed histories and next-event prediction instances. Given events through index i, a model predicts the value of event $i+1$. In setups with timing prediction, the model also predicts the elapsed time until event $i + 1$. For Time-IMM timing metrics, negative decoded interval predictions are retained rather than clipped, so impossible predicted next-event times are penalized directly. On the Time-IMM test sets, the validation-selected TSFM configurations using the event-sequence interface produce 673,270 decoded interval predictions. Of these, 55 are negative and none are NaN.

EasyTPP instances. For EasyTPP, we use the original event sequences from seven datasets. Each prediction instance contains the timing and type of the next event. The TSFM configurations represent event type with either one-hot dimensions or a frequency-ordered scalar encoding, as described in section B.4. All compared methods are scored by timing MAE and MSE in each dataset's original time units. For TSFM timing predictions, decoded intervals are clamped at zero before computing these errors. Because each EasyTPP instance also includes a discrete event type, we report event-type error for the same methods.

Compared method groups. The paper compares three sets of methods, each tied to a different empirical question.

1. **Event-sequence versus regular-grid TSFM interfaces.** Time-IMM histories are exposed to the same frozen TSFMs either through the event-sequence interface with raw/log interval encodings or through regular-grid interface variants. Models are evaluated on next-event value prediction; for regular-grid interfaces, the forecast is read at the first future grid step at or after the true next-event timestamp. We report normalized value MAE and MSE.

2. **Event-sequence TSFM interface versus trained irregular baselines.** Each event-sequence interface configuration specifies a frozen TSFM and an interval encoding. We compare these configurations with irregular time-series models trained on each dataset's training split and evaluated on the same next-event prediction instances. The trained irregular-baseline comparison uses two setups: joint timing prediction and oracle time. In the joint timing prediction setup, we attach a scalar interval-prediction head to each trained irregular baseline. The predicted interval is added to the most recent observed timestamp to obtain the query time for the next-value forecast. Training uses both value and interval-regression losses. In the oracle time setup, the model instead receives the ground-truth next-event timestamp and predicts only the value at that timestamp. The oracle time setup therefore has no timing-error metric. Trained irregular baselines under joint timing prediction are scored on both next-event value and next-event timing, while the same baselines under oracle time are scored on next-event value only. We report normalized value MAE and MSE, and relative timing MAE and MSE.

3. **EasyTPP timing and event-type results.** TSFM configurations are compared with trained TPP baselines on timing prediction. We report timing MAE and MSE in each dataset's original time units. For the same timing-selected methods, we also report event-type error.

C.2 Metrics and Normalization

Following the released Time-IMM evaluation pipeline, Time-IMM value errors are reported on the normalized value scale. Following the released EasyTPP evaluation pipeline, EasyTPP timing errors are reported directly in each dataset's native time unit. The released IMM-TSF benchmark library accompanying Time-IMM (Chang et al., 2025) standardizes each feature column before computing MSE and MAE, which is equivalent to scaling value errors by the feature standard deviation. For our next-event Time-IMM aggregation, we therefore report value errors after scaling each value dimension by its series-level standard deviation, so cross-series averages are not dominated by high-variance dimensions. Let $\mathcal{M}_{i+1}$ be the evaluated value dimensions for event $i + 1$, and let $s_j = \mathrm{std}(\{x_{\ell,j}\}_\ell)$ be the full-series standard deviation of dimension j. Dimensions that are missing under the dataset mask are excluded from $\mathcal{M}_{i+1}$. Dimensions with $s_j \leq 10^{-6}$ are also excluded, because

their normalized errors would be unstable. The normalized value absolute error for one prediction instance is

$$e^{\text{val}}_{\text{MAE}}(i+1) = \frac{1}{|\mathcal{M}_{i+1}|} \sum_{j \in \mathcal{M}_{i+1}} \frac{|\hat{x}_{i+1,j} - x_{i+1,j}|}{s_j}.$$

The normalized value squared error uses the same dimension scales,

$$e^{\text{val}}_{\text{MSE}}(i+1) = \frac{1}{|\mathcal{M}_{i+1}|} \sum_{j \in \mathcal{M}_{i+1}} \left(\frac{\hat{x}_{i+1,j} - x_{i+1,j}}{s_j} \right)^2.$$

Because our next-event evaluation also predicts event timing, Time-IMM timing error is reported relative to each series' timing scale. For each series, the timing scale is

$$m_{\Delta t} = \max\big(\text{median}\left(\{|\Delta t_j| : \Delta t_j \text{ is evaluated}\}\right), 10^{-6}\big).$$

The relative timing absolute error for one prediction instance is

$$e^{\text{time}}_{\text{MAE}}(i+1) = \frac{|\hat{t}_{i+1} - t_{i+1}|}{m_{\Delta t}}, \qquad \hat{t}_{i+1} = t_i + \Delta\hat{t}_{i+1}.$$

The relative timing squared error uses the same timing scale,

$$e^{\text{time}}_{\text{MSE}}(i+1) = \left(\frac{\hat{t}_{i+1} - t_{i+1}}{m_{\Delta t}} \right)^2.$$

For each Time-IMM next-event prediction, we first compute the normalized error. We then average these errors within each series, and dataset-family summaries average the resulting series-level errors within the family. The value and timing normalizations are applied only when computing the reported errors. For EasyTPP, let t_q and $\hat{t}_q$ be the true and predicted next-event times for prediction instance q, expressed in the dataset's native time unit. For a dataset with prediction instances $\mathcal{Q}$, we report

$$e^{\text{EasyTPP}}_{\text{MAE}} = \frac{1}{|\mathcal{Q}|} \sum_{q \in \mathcal{Q}} |\hat{t}_q - t_q|, \qquad e^{\text{EasyTPP}}_{\text{MSE}} = \frac{1}{|\mathcal{Q}|} \sum_{q \in \mathcal{Q}} (\hat{t}_q - t_q)^2.$$

EasyTPP timing comparisons are reported within each dataset. Event-type error is the fraction of prediction instances whose decoded event type does not match the next event type.

C.3 Regular-Grid Variants and Output-Budget Selection

For each Time-IMM prediction instance, each regular-grid variant converts the observed history into a grid sequence. The value forecast is read from the first future grid step at or after the true next-event timestamp.

- **Sample-and-hold.** Each grid position uses the latest observed value at or before the grid time.
- **Linear interpolation.** Values at unobserved grid positions are linearly interpolated between neighboring observed events.
- **Zero fill.** Grid ticks that coincide with an observed event carry that event value, and other grid ticks are filled with zero.
- **Observation mask.** Each value dimension uses sample-and-hold, and additional binary mask dimensions indicate which grid ticks coincide with observed events. We evaluate this variant only for TSFM input formats that support the added mask dimensions; unsupported TSFMs therefore do not report observation-mask results. In our implementation, this variant is evaluated for the multivariate-input TSFMs Chronos-2, Toto 1.0, and Toto 2.0, and not for the univariate-input TSFMs CTSM 1.0 and TimesFM 2.5.

For the validation-selected regular-grid comparison in table 1, regular-grid configurations are evaluated with output budgets of 64, 128, and 256 future grid steps. For each budget, grid spacing is selected separately for each Time-IMM series from validation-split next-event gaps. Starting from the greatest common divisor of the observed positive timestamp gaps, the spacing is doubled until

the largest validation next-event gap is reachable within that output budget. Validation normalized value MAE then selects the regular-grid variant and output budget for each dataset family and TSFM; held-out test metrics use the selected configuration. This avoids test-set horizon selection while retaining the regular-grid resolution tradeoff: smaller spacing is more precise but consumes more forecast positions, while larger spacing reaches longer gaps but coarsens timing. The validation-selected complete tables in section F.1 and the output-budget sensitivity tables in section D.1 use the same readout rule. For each prediction instance, the history grid is anchored backward from the last observed timestamp, and future ticks start one grid spacing later.

Zero fill and observation masks use the same series-specific grid. Zero fill sets non-observation ticks to zero; the mask variant uses sample-and-hold values plus mask channels indicating which ticks coincide with observations.

C.4 Trained Irregular Time-Series Baselines

For the Time-IMM trained-baseline comparison in section 4.3, we use CRU, Latent ODE, Neural Flow, and tPatchGNN, each trained on the corresponding dataset training split. This set is chosen for two reasons. First, it spans complementary mechanisms for irregular forecasting: continuous-time recurrence, latent ODE dynamics, neural-flow dynamics, and patch/graph modeling. Second, the reported Time-IMM unimodal results do not identify a single uniformly strongest irregular baseline among these methods. CRU is strongest among the four on CESNET and ILINet, Latent ODE is strongest on FNSPID, and Neural Flow is strongest on GDELT by MSE. tPatchGNN is strongest on ClusterTrace, EPA-Air, MIMIC, RepoHealth, and StudentLife (Chang et al., 2025). Our trained irregular baselines build on the implementations and model-specific hyperparameters in the released IMM-TSF benchmark library accompanying Time-IMM (Chang et al., 2025). We adapt these implementations to the next-event task using the training objectives and interval-prediction heads described below.

- **CRU** is a continuous-time recurrent model for irregularly sampled sequences (Schirmer et al., 2022). It updates a latent recurrent state at observation times and propagates that state across variable time gaps, so elapsed time is part of the state transition rather than a nuisance to be imputed away. In Time-IMM, CRU is configured with 32 latent and hidden units, square/exponential activations, and gravity gates, and it is the strongest of the four irregular baselines on CESNET and ILINet in the unimodal detailed results (Chang et al., 2025).
- **Latent ODE** encodes an irregular observation history into a latent initial state and evolves the latent trajectory with an ordinary differential equation before decoding values at requested query times (Rubanova et al., 2019). This makes the forecast function explicitly continuous in time and naturally supports predictions at nonuniform future timestamps. Time-IMM configures it with a 32-unit recognition GRU, one RNN layer, one generator layer, and 32 ODE units; among the four irregular baselines, it is strongest on FNSPID and is often strong across datasets (Chang et al., 2025).
- **Neural Flow** replaces numerical ODE integration with a flow-based continuous-time latent transformation (Biloš et al., 2021). A recurrent encoder summarizes the observed sequence, while coupling-flow layers and a time-conditioned network map latent states across query times, providing a different inductive bias from both recurrent updates and ODE solvers. Time-IMM configures it with a 20-dimensional latent state, a three-layer hidden network, 32 GRU units, two coupling-flow layers, and a TimeLinear time network; it is the strongest of the four irregular baselines on GDELT by unimodal MSE (Chang et al., 2025).
- **tPatchGNN** models irregular multivariate histories through transformable temporal patches and graph-based cross-variable interactions (Zhang et al., 2024). The patch representation organizes sparse observations into local temporal units, while Transformer and graph layers capture temporal context and dependencies among variables. Time-IMM configures it with patch size 24, one Transformer layer, one GNN layer, 10-dimensional time and node embeddings, and hidden dimension 32; among the four irregular baselines, it is strongest on the largest number of Time-IMM datasets (Chang et al., 2025).

Each baseline is trained separately on the training split of each dataset and evaluated on the same next-event value-and-timing prediction instances used in the main Time-IMM comparison. For

joint timing prediction, we attach a two-layer scalar interval-prediction head to each model's final hidden or latent context. Let z be the scalar output of this head, and define the predicted interval as $\Delta\hat{t} = \mathrm{softplus}(z) + 10^{-6}$ seconds. For raw interval targets, the interval loss compares $\Delta\hat{t}$ with Δt; for log-transformed interval targets, it compares $\log(1 + \Delta\hat{t}/\tau_0)$ with $\log(1 + \Delta t/\tau_0)$. Evaluation also uses $\Delta\hat{t}$ in seconds. The value normalizer is fit on the training split. The value loss is masked MSE on train-normalized values, and the interval loss is MSE on the chosen interval target with weight 1.0. In oracle time, the model sets the interval-loss weight to 0 and receives the ground-truth next-event timestamp as the query time for the value decoder. All runs use Adam with learning rate 10^{-3}, weight decay 0.01, and batch size 32; CRU additionally uses gradient clipping with norm 1.0. The initial training budget is 50 epochs with validation-loss checkpointing and early stopping patience 4. Runs whose best-validation-loss checkpoint reached the final epoch were continued with the same optimizer hyperparameters until early stopping or a maximum reported budget of 500 epochs. Within each trained run, checkpoint selection uses validation loss. Test metrics are used only for final reporting.

C.5 Trained Temporal Point Process Baselines

For the EasyTPP timing and event-type comparison in section 4.4, we use NHP, THP, AttNHP, and IFTPP as TPP baselines. EasyTPP evaluates one classical Hawkes baseline and eight neural TPPs, and its next-event prediction results indicate that no single method dominates across datasets (Xue et al., 2024). We choose these four baselines because they are repeatedly among the stronger learned TPP methods in EasyTPP's next-event results. NHP is strongest on Retweet and Taxi. THP and AttNHP represent attention-based methods that are strong or close on Amazon, Taobao, and StackOverflow. IFTPP contributes an intensity-free baseline with strong results on Amazon time prediction and Retweet type prediction (Xue et al., 2024). They also cover complementary temporal point process families:

- **NHP** is a recurrent, intensity-based neural Hawkes process whose continuous-time hidden state parameterizes event-type conditional intensities (Mei & Eisner, 2017). Observed events update the recurrent state, while the state evolves between events, allowing the model to represent history-dependent excitation without a fixed parametric Hawkes kernel. In the EasyTPP results, NHP is the strongest method on Retweet and Taxi and remains comparable to attention-based TPPs (Xue et al., 2024). The NHP configuration uses hidden size 64, time-embedding size 16, and two layers.
- **THP** replaces recurrent history compression with Transformer-style self-attention over the event history (Zuo et al., 2020). Event embeddings and temporal encodings produce contextual representations from which the model parameterizes the conditional intensity, so long-range dependencies can be routed through attention heads rather than a single recurrent state. This makes THP the direct Transformer-family TPP baseline; EasyTPP reports it among the strongest attention-based methods, including the best Taobao type-error result (Xue et al., 2024). The THP configuration uses hidden size 64, time-embedding size 16, two layers, and two attention heads.
- **AttNHP** is an attentive neural Hawkes process that builds attention-based representations for irregularly spaced event histories (Mei et al., 2022). It keeps the marked point-process objective while letting predictions attend over previous events, giving a different inductive bias from both recurrence-only NHP and the direct Transformer Hawkes formulation. The comparison uses AttNHP to represent attention-augmented Hawkes models; EasyTPP reports leading results for Amazon type prediction and Taobao/StackOverflow time prediction (Xue et al., 2024). The AttNHP configuration uses hidden size 32, time-embedding size 16, one layer, and two attention heads.
- **IFTPP** is an intensity-free temporal point process baseline that directly models the conditional distribution of elapsed time until the next event instead of parameterizing an intensity and approximating its integral (Shchur et al., 2020). This gives closed-form likelihood evaluation in EasyTPP's analysis, whereas intensity-based baselines require Monte Carlo integration for the intensity term (Xue et al., 2024). The comparison uses IFTPP as the intensity-free representative; EasyTPP reports strong next-event prediction results for Amazon timing and Retweet type error (Xue et al., 2024). The IFTPP configuration uses hidden size 32, time-embedding size 16, one layer, and three mixture components;

the median of its learned next-time distribution is used as the deterministic timing point estimate.

This set covers recurrent intensity modeling, Transformer intensity modeling, attention-augmented Hawkes modeling, and intensity-free distribution modeling in the EasyTPP timing and event-type comparison. All TPP baselines are trained with the released EasyTPP training pipeline.

D Additional Supporting and Sensitivity Results

This section collects supporting analyses for the main Time-IMM and EasyTPP result tables: regular-grid output-budget sensitivity, family-level paired gaps, interval-encoding sensitivity, cross-TSFM comparison, interval-history perturbation controls, and EasyTPP event-type representation.

D.1 Time-IMM Regular-Grid Output-Budget Sensitivity

Tables 6 and 7 report Time-IMM test errors by dataset family. The shortened dataset headers are CES = CESNET, Cluster = ClusterTrace, EPA = EPA-Air, ILI = ILINet, Repo = RepoHealth, and Student = StudentLife; FNSPID, GDELT, and MIMIC are unchanged. For the event-sequence entries, validation MAE selects the raw/log encoding in table 6, and validation MSE selects the raw/log encoding in table 7. For the regular-grid entries, the same validation metric selects the regular-grid variant for each of the three output budgets (64, 128, and 256 future grid steps). These tables show the budget-level evaluation results behind the aggregate interface comparison in the main text. Across output budgets, the aggregate comparison continues to favor the event-sequence interface for every evaluated TSFM. Averaged over all series/TSFM pairs, event-sequence MAE is 0.424, while regular-grid MAE is 0.472, 0.476, and 0.495 for 64-, 128-, and 256-step output budgets; event-sequence MSE is 0.608, while regular-grid MSE is 0.771, 0.971, and 1.442. Thus, the event-sequence interface achieves lower average value error than regular-grid interfaces across the tested output budgets.

Table 6: Time-IMM interface test MAE by dataset family and regular-grid budget. Lower is better.

TSFM	Interface	Budget	CES	Cluster	EPA	FNSPID	GDELT	ILI	MIMIC	Repo	Student
Chronos-2	Event sequence	–	0.696	0.325	0.118	0.092	0.611	0.196	0.399	0.257	0.571
	Regular grid	64	0.790	0.392	0.156	0.092	0.699	0.207	0.449	0.250	0.568
	Regular grid	128	0.828	0.390	0.118	0.092	0.714	0.207	0.488	0.250	0.568
	Regular grid	256	0.869	0.392	0.118	0.092	0.731	0.207	0.588	0.250	0.568
CTSM 1.0	Event sequence	–	0.705	0.323	0.121	0.094	0.646	0.205	0.371	0.267	0.601
	Regular grid	64	0.758	0.406	0.153	0.095	0.700	0.206	0.440	0.250	0.600
	Regular grid	128	0.755	0.408	0.120	0.095	0.708	0.206	0.409	0.250	0.600
	Regular grid	256	0.764	0.409	0.120	0.095	0.700	0.206	0.378	0.250	0.600
TimesFM 2.5	Event sequence	–	0.696	0.329	0.121	0.092	0.615	0.195	0.373	0.262	0.587
	Regular grid	64	0.802	0.408	0.153	0.092	0.695	0.198	0.468	0.255	0.585
	Regular grid	128	0.824	0.409	0.120	0.092	0.707	0.198	0.501	0.255	0.585
	Regular grid	256	0.843	0.416	0.120	0.092	0.714	0.198	0.598	0.255	0.585
Toto 1.0	Event sequence	–	0.705	0.345	0.117	0.091	0.618	0.195	0.372	0.264	0.601
	Regular grid	64	0.850	0.414	0.150	0.092	0.739	0.198	0.458	0.267	0.601
	Regular grid	128	0.855	0.417	0.116	0.092	0.751	0.198	0.485	0.267	0.601
	Regular grid	256	0.882	0.425	0.116	0.092	0.756	0.198	0.579	0.267	0.601
Toto 2.0	Event sequence	–	0.684	0.298	0.113	0.091	0.609	0.174	0.360	0.251	0.594
	Regular grid	64	0.848	0.430	0.158	0.094	0.743	0.194	0.465	0.251	0.578
	Regular grid	128	0.854	0.433	0.123	0.094	0.755	0.194	0.489	0.251	0.578
	Regular grid	256	0.862	0.437	0.123	0.094	0.754	0.194	0.576	0.251	0.578

D.2 Time-IMM Family-Level Error Gaps

Tables 1 and 2 summarize aggregate Time-IMM performance by dataset family. Table 8 gives the family-level paired version of those Time-IMM comparisons. For each row, errors are averaged within each dataset family for the TSFM using the event-sequence interface and the listed comparator. The reported gap is the comparator error minus the event-sequence error. Thus, a positive gap favors the event-sequence interface. The "Wins" column counts how many of the nine Time-IMM

Table 7: Time-IMM interface test MSE by dataset family and regular-grid budget. Lower is better.

TSFM	Interface	Budget	CES	Cluster	EPA	FNSPID	GDELT	ILI	MIMIC	Repo	Student
Chronos-2	Event sequence	–	0.835	0.197	0.091	0.062	1.048	0.138	0.713	0.434	0.732
	Regular grid	64	1.080	0.242	0.250	0.061	1.288	0.148	0.991	0.419	0.722
	Regular grid	128	1.190	0.241	0.090	0.061	1.344	0.148	2.110	0.419	0.722
	Regular grid	256	1.310	0.242	0.090	0.061	1.418	0.148	4.630	0.419	0.722
CTSM 1.0	Event sequence	–	0.818	0.197	0.106	0.061	1.005	0.151	0.721	0.416	0.754
	Regular grid	64	0.994	0.240	0.227	0.062	1.312	0.154	0.920	0.399	0.751
	Regular grid	128	0.984	0.243	0.106	0.062	1.361	0.154	0.863	0.399	0.751
	Regular grid	256	1.001	0.244	0.106	0.062	1.339	0.154	0.755	0.399	0.751
TimesFM 2.5	Event sequence	–	0.831	0.195	0.107	0.062	1.033	0.152	0.734	0.437	0.751
	Regular grid	64	1.124	0.247	0.228	0.061	1.278	0.154	1.164	0.425	0.744
	Regular grid	128	1.184	0.245	0.106	0.061	1.321	0.154	2.255	0.425	0.744
	Regular grid	256	1.243	0.251	0.106	0.061	1.341	0.154	4.742	0.425	0.744
Toto 1.0	Event sequence	–	0.876	0.204	0.102	0.064	1.109	0.157	0.767	0.451	0.810
	Regular grid	64	1.291	0.250	0.223	0.062	1.471	0.163	1.160	0.461	0.807
	Regular grid	128	1.304	0.251	0.101	0.062	1.536	0.163	2.230	0.461	0.807
	Regular grid	256	1.374	0.261	0.101	0.062	1.570	0.163	4.692	0.461	0.807
Toto 2.0	Event sequence	–	0.811	0.229	0.092	0.061	1.045	0.128	0.775	0.430	0.789
	Regular grid	64	1.278	0.266	0.231	0.061	1.507	0.147	1.199	0.431	0.744
	Regular grid	128	1.295	0.269	0.109	0.061	1.592	0.147	2.266	0.431	0.744
	Regular grid	256	1.329	0.275	0.109	0.061	1.577	0.147	4.708	0.431	0.744

dataset families have lower error for frozen TSFMs using the event-sequence interface than for the listed comparator.

Table 8: Family-level Time-IMM error gaps for frozen TSFMs using the event-sequence interface against regular-grid interfaces and trained irregular baselines. Configurations are selected by validation value MAE.

Comparator	Metric	Wins	Mean gap
Regular grid	Value MAE	8/9	0.0488
Regular grid	Value MSE	8/9	0.1384
Irregular baseline (joint timing)	Value MAE	9/9	0.2085
Irregular baseline (joint timing)	Value MSE	8/9	0.2248
Irregular baseline (joint timing)	Timing MAE	9/9	0.8376
Irregular baseline (joint timing)	Timing MSE	9/9	16.5206
Irregular baseline (oracle time)	Value MAE	9/9	0.2558
Irregular baseline (oracle time)	Value MSE	8/9	0.3205

The first two rows summarize the comparison between event-sequence and regular-grid interfaces in table 1. Frozen TSFMs using the event-sequence interface have lower value MAE and MSE in 8 of 9 families, and the mean gaps for both metrics are positive. The remaining rows summarize comparisons with trained irregular baselines under joint timing prediction and oracle time in table 2. Under joint timing prediction, frozen TSFMs using the event-sequence interface have lower value and timing MAE and MSE in all nine families. Under oracle time, they have lower value MAE in all nine families and lower value MSE in eight.

D.3 Raw Versus Log-Transformed Intervals

Table 9 compares raw and log-transformed interval encodings for each TSFM. Each cell reports Raw/Tie/Log counts: the first number is the number of cases where raw intervals give lower error, the middle number is the number of ties, and the last number is the number of cases where log-transformed intervals give lower error. Time-IMM counts use the nine dataset families; EasyTPP counts use the seven datasets. For EasyTPP, validation timing MAE first selects the event-type encoding separately for each raw/log interval choice.

The table shows that the preferred encoding can change between value error, timing error, and event-type error, and also varies by benchmark and TSFM. For Time-IMM, raw intervals more often lower timing MAE/MSE; for value MAE/MSE, neither raw nor log consistently dominates across TSFMs. For EasyTPP, the preferred raw/log encoding also depends on the TSFM for both timing error and

Table 9: Raw/log interval-encoding sensitivity. Lower error indicates the preferred encoding. EasyTPP type-error counts exclude Volcano because it has one event type.

TSFM	Time-IMM				EasyTPP		
	Value MAE	Value MSE	Timing MAE	Timing MSE	Timing MAE	Timing MSE	Type Err.
Chronos-2	2/0/7	1/0/8	5/0/4	5/0/4	4/0/3	6/0/1	2/0/4
CTSM 1.0	0/9/0	0/9/0	5/0/4	8/0/1	1/0/6	1/0/6	0/6/0
TimesFM 2.5	0/9/0	0/9/0	5/0/4	8/0/1	1/0/6	2/0/5	0/6/0
Toto 1.0	5/0/4	4/0/5	4/0/5	5/0/4	5/0/2	5/0/2	3/0/3
Toto 2.0	5/0/4	4/0/5	7/0/2	6/0/3	2/0/5	2/0/5	2/0/4

event-type error. The main comparisons therefore choose raw or log intervals separately for each dataset and TSFM using validation data.

D.4 Cross-TSFM Comparison

The next tables compare TSFM families under the main configuration-selection rule. For Time-IMM, validation normalized value MAE selects the raw/log interval encoding for each dataset and TSFM. For EasyTPP, validation timing MAE selects the raw/log interval encoding and event-type encoding for each dataset and TSFM.

Table 10: Time-IMM comparison across TSFMs. Cells show test value MAE with selected encoding: R = raw, L = log. Lower is better.

Dataset	Chronos-2	CTSM 1.0	TimesFM 2.5	Toto 1.0	Toto 2.0
CESNET	0.696 (L)	0.705 (R)	0.696 (R)	0.706 (L)	**0.684 (R)**
ClusterTrace	0.326 (R)	0.323 (R)	0.329 (R)	0.345 (R)	**0.298 (L)**
EPA-Air	0.118 (L)	0.121 (R)	0.121 (R)	0.117 (R)	**0.113 (R)**
FNSPID	0.092 (L)	0.094 (R)	0.092 (R)	0.091 (L)	**0.091 (L)**
GDELT	0.611 (L)	0.646 (R)	0.615 (R)	0.618 (L)	**0.609 (L)**
ILINet	0.197 (R)	0.205 (R)	0.195 (R)	0.195 (R)	**0.174 (L)**
MIMIC	0.399 (L)	0.371 (R)	0.373 (R)	0.372 (R)	**0.360 (R)**
RepoHealth	0.257 (L)	0.267 (R)	0.262 (R)	0.264 (L)	**0.251 (L)**
StudentLife	**0.571 (R)**	0.601 (R)	0.587 (R)	0.601 (L)	0.594 (R)
Macro avg.	0.363	0.370	0.363	0.368	**0.353**
Avg. rank	2.89	4.22	3.22	3.44	**1.22**

Table 11: EasyTPP comparison across TSFMs. Cells show test timing MAE with selected interval encoding: R = raw, L = log. The summary row reports average rank. Lower is better.

Dataset	Chronos-2	CTSM 1.0	TimesFM 2.5	Toto 1.0	Toto 2.0
Amazon	0.269 (R)	0.294 (R)	**0.262 (R)**	0.293 (R)	0.285 (R)
Earthquake	0.800 (R)	0.763 (L)	**0.761 (L)**	0.865 (R)	0.802 (L)
Retweet	2353 (L)	2331 (L)	**2326 (L)**	2352 (L)	2333 (L)
StackOverflow	0.696 (R)	0.705 (L)	**0.687 (L)**	0.730 (R)	0.702 (L)
Taobao	0.047 (L)	0.059 (L)	0.045 (L)	0.046 (L)	**0.045 (L)**
Taxi	0.164 (L)	0.172 (L)	**0.162 (L)**	0.175 (R)	0.166 (L)
Volcano	1403 (L)	**953.1 (L)**	992.7 (L)	1216 (R)	1103 (R)
Avg. rank	3.29	3.29	**1.29**	4.29	2.86

Table 10 shows that Time-IMM value MAE is close across TSFMs, ranging from 0.353 to 0.370. Toto 2.0 has the best average rank across dataset families, but the differences among TSFMs are small. Table 11 shows that EasyTPP timing results vary by TSFM and dataset. TimesFM 2.5 has the best average rank across datasets in this comparison. Grouped by input format, the multivariate-input TSFMs have a better average rank on Time-IMM, whereas the univariate-input TSFMs have a better average rank on EasyTPP. This descriptive pattern does not isolate an input-format effect because the models also differ in architecture and pretraining. Thus, the TSFM ordering is benchmark- and aggregation-dependent; the common signal is that the event-sequence interface works consistently across multiple TSFM families.

D.5 Interval-History Perturbation Controls

This subsection tests the validation-selected configurations of frozen TSFMs using the event-sequence interface in the Time-IMM and EasyTPP experiments. For each held-out prediction instance, we keep the same frozen TSFM and target, but perturb part of the event-sequence input history before forecasting. The controls shuffle or zero interval histories. Table 12 reports how these interval-history perturbations change next-event timing error on Time-IMM and EasyTPP. Gaps compare the perturbed input with the original event-sequence input for the same frozen TSFM. Time-IMM rows report both mean/median gaps in relative timing MAE and direction counts across nine dataset families. EasyTPP uses direction counts only because its timing MAE is measured in each dataset's own time unit; the direction column counts cases where the perturbed input has worse timing MAE than the original input, and EasyTPP counts seven datasets.

Table 12: Interval-history perturbation controls for next-event timing.

Benchmark	Control	Metric	Mean gap	Median gap	Direction
Time-IMM	Shuffled intervals	Relative timing MAE	+0.241	+0.083	8/9 families worse
Time-IMM	Zero intervals	Relative timing MAE	+0.866	+0.984	9/9 families worse
EasyTPP	Shuffled intervals	Timing MAE	–	–	6/7 datasets worse
EasyTPP	Zero intervals	Timing MAE	–	–	5/7 datasets worse

Table 12 shows that the evaluated TSFMs' next-event timing predictions depend on the supplied interval history. Shuffling or zeroing the interval history usually increases error: Time-IMM worsens in 8/9 and 9/9 families, and EasyTPP worsens in 6/7 and 5/7 datasets.

D.6 EasyTPP Event-Type Representation

The EasyTPP timing and event-type comparison requires a numeric representation of event types for TSFMs, so we compare one-hot and frequency-ordered scalar encodings.

Table 13: EasyTPP event-type representation effects. Timing columns report average rank across seven datasets; type-error means and type counts exclude Volcano because it has one event type. Lower is better.

TSFM	Interval	One-hot			Frequency Rank			Type counts
		Timing MAE rank	Timing MSE rank	Type Err.	Timing MAE rank	Timing MSE rank	Type Err.	
Chronos-2	Raw	1.36	1.21	0.495	1.64	1.79	0.569	5/1/0
	Log	1.36	1.36	0.495	1.64	1.64	0.569	5/1/0
CTSM 1.0	Raw	1.57	1.50	0.512	1.43	1.50	0.629	6/0/0
	Log	1.50	1.50	0.512	1.50	1.50	0.629	6/0/0
TimesFM 2.5	Raw	1.50	1.50	0.492	1.50	1.50	0.574	6/0/0
	Log	1.50	1.50	0.492	1.50	1.50	0.574	6/0/0
Toto 1.0	Raw	1.86	1.71	0.543	1.14	1.29	0.535	2/4/0
	Log	1.86	2.00	0.543	1.14	1.00	0.536	2/4/0
Toto 2.0	Raw	1.29	1.43	0.505	1.71	1.57	0.532	4/2/0
	Log	1.14	1.14	0.505	1.86	1.86	0.533	4/2/0

Table 13 shows that event-type representation has a clearer effect on type prediction than on timing prediction. One-hot encoding usually gives lower event-type error. For timing, neither representation is consistently better; the ranking depends on the TSFM family.

E Dataset Characteristics Behind Time-IMM Performance Gaps

This appendix uses Time-IMM dataset characteristics to explain performance gaps among frozen TSFMs using the event-sequence interface, the same frozen TSFMs using regular-grid interfaces, and trained irregular baselines.

First consider the event-sequence and regular-grid interfaces for the same frozen TSFMs. Both interfaces are scored on the ground-truth next-event value, but they place that value at different

forecast positions. The event-sequence interface makes the next observed event the next sequence position and forecasts elapsed time as a numeric coordinate. The regular-grid interface instead forecasts future clock ticks and reads out the first tick at or after the true next-event time. This regular-grid readout introduces two potential sources of error. If many grid ticks lie before the next event, the model spends output positions on times with no observed event. If the value changes sharply between observed events, sample-and-hold or interpolation can place grid values far from the value eventually scored at the event. Because the event-sequence interface forecasts directly at the next observed event, it avoids both potential sources of error.

Table 14 reports the value-error gaps and the two dataset characteristics associated with these potential sources of error. The value gaps are regular-grid error minus event-sequence error, so positive values favor the event-sequence interface. They use the validation value-MAE selection rule and are averaged over paired test comparisons within each family. "Grid steps to target" is the average number of future grid ticks needed to reach the true next event, computed as $\lceil (t_{i+1} - t_i)/h \rceil$ for the selected grid spacing h. "Value change" measures how much the observed value usually moves from one event to the next; it is the average normalized one-step value change, $|x_{i+1,c} - x_{i,c}|/s_c$, over test targets and nonconstant numeric channels, then averaged over series in the family.

Table 14: All-family regular-grid gaps and diagnostics.

Family	Value MAE gap	Value MSE gap	Grid steps to target	Value change
CESNET	0.112	0.319	17.67	0.927
GDELT	0.096	0.343	7.742	0.827
ClusterTrace	0.091	0.047	3.762	0.447
MIMIC	0.069	0.314	22.51	0.539
EPA-Air	0.039	0.141	1.102	0.109
RepoHealth	0.026	0.080	2.049	0.344
ILINet	0.009	0.011	1.000	0.228
FNSPID	0.001	0.000	1.447	0.102
StudentLife	-0.003	-0.008	1.004	0.741

Table 14 shows the same qualitative pattern for MAE and MSE: the value gaps are positive for almost every family and therefore favor frozen TSFMs using the event-sequence interface. StudentLife is the near-tie. CESNET and MIMIC have long grid readouts, requiring 17.7 and 22.5 future grid steps on average. GDELT has a shorter readout but large normalized value changes, so filled grid values can differ substantially from the next event value. By contrast, StudentLife and FNSPID reach the target in about one grid step on average; FNSPID also has small value changes, and StudentLife shows that large value changes alone do not create a large event-sequence gain when the selected grid lands close to the next event. The pattern is that the event-sequence interface helps most when the grid adds many empty forecast positions or when values move substantially between events. It offers less advantage when the selected grid already lands close to the next event and simple fill or interpolation is already close to the target value.

Whereas the preceding analysis focuses on sources of error introduced by the regular-grid interface, the trained-baseline analysis compares frozen pretrained TSFMs using the event-sequence interface with irregular models trained on Time-IMM. Under joint timing prediction, the baselines predict both the next timestamp and value; under oracle time, they receive the ground-truth next timestamp and predict the value. Unlike the baselines, the frozen TSFMs do not adapt their parameters to the Time-IMM training data. Through the event-sequence interface, they process recent values directly in observation order. When this ordering is informative about the next value, frozen TSFMs can perform well without dataset-specific adaptation. We therefore examine whether the performance gaps are associated with how strongly recent event values predict the next value.

Table 15 reports every Time-IMM family for the trained-baseline comparison. Each gap compares the error of a trained irregular baseline with the error of a validation-selected event-sequence interface configuration. Gaps are averaged over paired test comparisons within each family. Positive gaps mean lower error for the validation-selected event-sequence interface configuration; negative gaps mean lower error for the trained irregular baseline. Each gap cell reports joint timing prediction followed by oracle time. The last three columns summarize one-step value dynamics and value-order predictability. "Value change" measures the typical size of a one-step value update, after normalizing each channel by its series-level standard deviation. "95th/mean change" measures whether the family has rare large jumps: for each series, it is the 95th percentile of normalized one-step value

changes divided by their mean, then averaged over series in the family. "Value-order frac." measures value-order predictability: for each series, it is the fraction of eligible value channels whose observed autocorrelation exceeds the 90th percentile of shuffled value sequences, then averaged over series in the family, following section B.5.

Table 15: All-family trained-baseline gaps and diagnostics.

Family	MAE gap (joint/oracle)	MSE gap (joint/oracle)	Value change	95th/mean change	Value-order frac.
FNSPID	0.460 / 0.628	0.498 / 0.821	0.102	4.329	1.000
RepoHealth	0.292 / 0.391	0.278 / 0.446	0.344	4.377	0.925
MIMIC	0.262 / 0.262	0.248 / 0.286	0.539	4.508	0.929
ILINet	0.255 / 0.326	0.611 / 0.819	0.228	3.839	1.000
EPA-Air	0.231 / 0.314	0.161 / 0.246	0.109	2.548	1.000
ClusterTrace	0.159 / 0.185	0.075 / 0.148	0.447	2.747	1.000
StudentLife	0.091 / 0.080	0.080 / 0.070	0.741	2.900	0.802
CESNET	0.078 / 0.066	0.104 / 0.086	0.927	2.646	0.527
GDELT	0.048 / 0.049	-0.032 / -0.037	0.827	3.254	0.250

Results for FNSPID, EPA-Air, RepoHealth, and MIMIC favor frozen TSFMs using the event-sequence interface over trained irregular baselines. FNSPID and EPA-Air have value-order fractions of 1.000 and small normalized one-step value changes (0.102 and 0.109), so the next value is typically a small update from recent events. RepoHealth and MIMIC have larger value changes, but their value-order fractions remain high (0.925 and 0.929), indicating that the observed value order still provides useful information about the next value. Together, these families support the hypothesis that frozen TSFMs benefit when recent event values are predictive. This advantage persists under oracle time: the trained baselines receive the ground-truth next timestamp, yet their value error remains higher than that of the frozen TSFMs on all four families.

By contrast, GDELT shows that this advantage does not extend to every metric. Its value-order fraction is only 0.250, its normalized one-step value change is large (0.827), and its 95th/mean change is 3.254. Recent event values are a weaker guide to the next value, and rare jumps can produce large errors. The frozen-TSFM configurations still achieve lower value MAE than the trained irregular baselines under both joint timing prediction and oracle time, but the trained baselines achieve lower value MSE in both setups. Because MSE gives more weight to large errors, the trained baselines achieve lower squared error on GDELT even though their average absolute error is higher.

F Complete Result Tables

This section collects the full numerical tables behind the compact main-text comparisons and the supporting and sensitivity summaries above.

F.1 Complete Time-IMM Interface Results

The following tables expand the Time-IMM interface comparison by dataset family. Event-sequence rows report the raw and log interval encodings separately. Each supported regular-grid variant is reported separately; within each variant, validation normalized value MAE selects an output budget of 64, 128, or 256 future grid steps. Section D.1 reports the corresponding output-budget sensitivity. Dataset-family averages are computed over the processed series in each family. These tables use the dataset headers defined in section D.1. Table 16 reports normalized value MAE, and table 17 reports normalized value MSE.

Table 16: Complete Time-IMM interface results for normalized value MAE. Error and rank are dataset-family averages. Lower is better.

TSFM	Interface	CES		Cluster		EPA		FNSPID		GDELT		ILI		MIMIC		Repo		Student	
		Error	Rank	Error	Rank	Error	Rank	Error	Rank	Error	Rank	Error	Rank	Error	Rank	Error	Rank	Error	Rank
Chronos-2	Event raw	0.696	2.00	0.326	1.00	0.120	2.00	0.092	2.00	0.611	2.00	0.197	2.00	0.400	2.00	0.258	4.00	0.571	5.00
	Event log	0.696	1.00	0.328	2.00	0.118	1.00	0.092	1.00	0.611	1.00	0.196	1.00	0.399	1.00	0.257	3.00	0.571	6.00
	Hold	0.802	4.00	0.421	4.00	0.156	3.00	0.093	3.00	0.706	4.00	0.207	3.00	0.450	3.00	0.250	1.00	0.568	1.00
	Zero fill	1.823	6.00	2.427	6.00	0.157	5.00	0.146	6.00	1.241	6.00	0.213	5.00	3.325	6.00	0.340	6.00	0.570	3.00
	Linear	0.876	5.00	0.435	5.00	0.156	4.00	0.093	4.00	0.777	5.00	0.207	4.00	1.113	5.00	0.259	5.00	0.569	2.00
	Mask	0.790	3.00	0.396	3.00	0.157	6.00	0.093	5.00	0.700	3.00	0.213	6.00	0.453	4.00	0.255	2.00	0.570	4.00
CTSM 1.0	Event raw	0.705	1.50	0.323	1.50	0.121	1.50	0.094	1.50	0.646	1.50	0.205	1.50	0.371	1.50	0.267	3.50	0.601	1.50
	Event log	0.705	1.50	0.323	1.50	0.121	1.50	0.094	1.50	0.646	1.50	0.205	1.50	0.371	1.50	0.267	3.50	0.601	1.50
	Hold	0.755	3.00	0.409	3.00	0.154	4.00	0.095	3.00	0.700	3.00	0.206	3.00	0.378	3.00	0.251	1.00	0.601	3.00
	Zero fill	1.804	5.00	1.944	5.00	0.155	5.00	0.198	5.00	1.207	5.00	0.212	5.00	3.248	5.00	0.363	5.00	0.603	5.00
	Linear	0.823	4.00	0.449	4.00	0.153	3.00	0.095	4.00	0.758	4.00	0.206	4.00	0.445	4.00	0.263	2.00	0.601	4.00
TimesFM 2.5	Event raw	0.696	1.50	0.329	1.50	0.121	1.50	0.092	1.50	0.615	1.50	0.195	1.50	0.373	1.50	0.262	3.50	0.587	4.50
	Event log	0.696	1.50	0.329	1.50	0.121	1.50	0.092	1.50	0.615	1.50	0.195	1.50	0.373	1.50	0.262	3.50	0.587	4.50
	Hold	0.802	3.00	0.416	3.00	0.154	4.00	0.092	3.00	0.695	3.00	0.199	4.00	0.468	3.00	0.256	1.00	0.585	1.00
	Zero fill	1.740	5.00	1.986	5.00	0.158	5.00	0.153	5.00	1.236	5.00	0.198	3.00	3.461	5.00	0.476	5.00	0.586	3.00
	Linear	0.868	4.00	0.454	4.00	0.153	3.00	0.093	4.00	0.756	4.00	0.199	5.00	1.109	4.00	0.262	2.00	0.586	2.00
Toto 1.0	Event raw	0.706	1.00	0.345	2.00	0.117	2.00	0.091	2.00	0.618	2.00	0.195	1.00	0.372	1.00	0.264	1.00	0.601	1.00
	Event log	0.706	2.00	0.345	1.00	0.117	1.00	0.091	1.00	0.618	1.00	0.195	2.00	0.372	2.00	0.264	2.00	0.601	2.00
	Hold	0.850	3.00	0.417	3.00	0.150	4.00	0.092	3.00	0.739	3.00	0.198	5.00	0.458	3.00	0.269	3.00	0.602	4.00
	Zero fill	1.316	6.00	1.962	6.00	0.153	6.00	0.123	6.00	1.243	6.00	0.204	6.00	3.353	6.00	0.413	6.00	0.602	3.00
	Linear	0.870	4.00	0.437	4.00	0.150	5.00	0.093	5.00	0.750	4.00	0.198	3.00	1.102	5.00	0.284	5.00	0.603	6.00
	Mask	0.963	5.00	0.612	5.00	0.150	3.00	0.092	4.00	0.811	5.00	0.198	4.00	1.049	4.00	0.273	4.00	0.602	5.00
Toto 2.0	Event raw	0.684	1.00	0.298	2.00	0.113	1.00	0.091	1.00	0.609	1.00	0.174	2.00	0.360	2.00	0.251	2.00	0.594	5.00
	Event log	0.684	2.00	0.298	1.00	0.113	2.00	0.091	2.00	0.609	2.00	0.174	1.00	0.360	1.00	0.251	1.00	0.594	6.00
	Hold	0.857	4.00	0.436	4.00	0.158	3.00	0.094	3.00	0.745	3.00	0.194	3.00	0.465	3.00	0.254	3.00	0.579	1.00
	Zero fill	1.157	6.00	2.033	6.00	0.160	6.00	0.105	6.00	1.243	6.00	0.227	6.00	3.310	6.00	0.293	6.00	0.581	3.00
	Linear	0.849	3.00	0.435	3.00	0.158	5.00	0.094	4.00	0.758	4.00	0.195	5.00	1.119	4.00	0.263	5.00	0.581	4.00
	Mask	0.968	5.00	1.081	5.00	0.158	4.00	0.096	5.00	0.874	5.00	0.194	4.00	1.291	5.00	0.259	4.00	0.580	2.00

Table 17: Complete Time-IMM interface results for normalized value MSE. Error and rank are dataset-family averages. Lower is better.

TSFM	Interface	CES		Cluster		EPA		FNSPID		GDELT		ILI		MIMIC		Repo		Student	
		Error	Rank	Error	Rank	Error	Rank	Error	Rank	Error	Rank	Error	Rank	Error	Rank	Error	Rank	Error	Rank
Chronos-2	Event raw	0.837	2.00	0.199	2.00	0.092	2.00	0.062	4.00	1.049	2.00	0.139	2.00	0.715	2.00	0.437	3.00	0.732	4.00
	Event log	0.835	1.00	0.197	1.00	0.091	1.00	0.062	3.00	1.048	1.00	0.138	1.00	0.713	1.00	0.434	2.00	0.732	5.00
	Hold	1.108	4.00	0.258	4.00	0.251	3.00	0.062	2.00	1.305	4.00	0.148	3.00	0.997	3.00	0.419	1.00	0.724	1.00
	Zero fill	4.391	6.00	7.281	6.00	0.253	5.00	0.158	6.00	2.797	6.00	0.158	6.00	49.16	6.00	0.630	6.00	0.729	3.00
	Linear	1.355	5.00	0.274	5.00	0.252	4.00	0.062	5.00	1.609	5.00	0.149	4.00	11.75	5.00	0.454	5.00	0.727	2.00
	Mask	1.080	3.00	0.245	3.00	0.269	6.00	0.061	1.00	1.292	3.00	0.155	5.00	1.001	4.00	0.438	4.00	0.732	6.00
CTSM 1.0	Event raw	0.818	1.50	0.197	1.50	0.106	1.50	0.061	1.50	1.005	1.50	0.151	1.50	0.721	1.50	0.416	2.50	0.754	1.50
	Event log	0.818	1.50	0.197	1.50	0.106	1.50	0.061	1.50	1.005	1.50	0.151	1.50	0.721	1.50	0.416	2.50	0.754	1.50
	Hold	0.984	3.00	0.244	3.00	0.228	4.00	0.063	4.00	1.339	3.00	0.154	3.00	0.755	3.00	0.400	1.00	0.754	3.00
	Zero fill	4.334	5.00	5.066	5.00	0.231	5.00	0.113	5.00	2.701	5.00	0.160	5.00	48.97	5.00	0.596	5.00	0.761	5.00
	Linear	1.178	4.00	0.293	4.00	0.227	3.00	0.063	3.00	1.497	4.00	0.154	4.00	0.965	4.00	0.430	4.00	0.756	4.00
TimesFM 2.5	Event raw	0.831	1.50	0.195	1.50	0.107	1.50	0.062	3.50	1.033	1.50	0.152	1.50	0.734	1.50	0.437	2.50	0.751	4.50
	Event log	0.831	1.50	0.195	1.50	0.107	1.50	0.062	3.50	1.033	1.50	0.152	1.50	0.734	1.50	0.437	2.50	0.751	4.50
	Hold	1.124	3.00	0.251	3.00	0.231	4.00	0.061	1.00	1.278	3.00	0.156	5.00	1.164	3.00	0.426	1.00	0.746	1.00
	Zero fill	4.084	5.00	4.894	5.00	0.236	5.00	0.080	5.00	2.774	5.00	0.154	3.00	49.58	5.00	1.002	5.00	0.750	3.00
	Linear	1.313	4.00	0.300	4.00	0.228	3.00	0.062	2.00	1.501	4.00	0.156	4.00	11.83	4.00	0.439	4.00	0.748	2.00
Toto 1.0	Event raw	0.876	2.00	0.204	2.00	0.102	2.00	0.064	5.00	1.110	2.00	0.157	1.00	0.767	1.00	0.451	1.00	0.810	1.00
	Event log	0.876	1.00	0.204	1.00	0.102	1.00	0.064	4.00	1.109	1.00	0.157	2.00	0.767	2.00	0.451	2.00	0.810	2.00
	Hold	1.291	3.00	0.251	3.00	0.223	3.00	0.062	1.00	1.482	3.00	0.163	5.00	1.160	3.00	0.480	3.00	0.810	3.00
	Zero fill	2.703	6.00	7.553	6.00	0.230	6.00	0.095	6.00	2.805	6.00	0.173	6.00	49.28	6.00	0.864	6.00	0.810	4.00
	Linear	1.369	4.00	0.276	4.00	0.223	5.00	0.063	3.00	1.543	4.00	0.163	3.00	11.82	4.00	0.545	5.00	0.814	6.00
	Mask	1.705	5.00	4.011	5.00	0.223	4.00	0.063	2.00	1.782	5.00	0.163	4.00	12.26	5.00	0.488	4.00	0.811	5.00
Toto 2.0	Event raw	0.811	1.00	0.229	2.00	0.092	1.00	0.061	2.00	1.045	1.00	0.128	2.00	0.776	1.00	0.430	2.00	0.789	6.00
	Event log	0.811	2.00	0.229	1.00	0.092	2.00	0.061	1.00	1.045	2.00	0.128	1.00	0.776	2.00	0.430	1.00	0.789	5.00
	Hold	1.308	4.00	0.269	3.00	0.231	3.00	0.062	5.00	1.562	3.00	0.148	5.00	1.199	3.00	0.455	3.00	0.747	1.00
	Zero fill	2.267	6.00	5.614	6.00	0.237	6.00	0.062	3.00	2.804	6.00	0.257	6.00	49.18	6.00	0.542	6.00	0.755	4.00
	Linear	1.286	3.00	0.270	4.00	0.231	5.00	0.062	4.00	1.682	4.00	0.147	4.00	11.98	4.00	0.488	5.00	0.751	2.00
	Mask	1.627	5.00	2.103	5.00	0.231	4.00	0.063	6.00	2.082	5.00	0.147	3.00	13.52	5.00	0.461	4.00	0.751	3.00

F.2 Complete Time-IMM Irregular-Model Results

The following tables provide complete Time-IMM results for the comparison between frozen TSFMs and trained irregular baselines. The frozen TSFMs use the event-sequence interface, and the trained baselines are CRU, Latent ODE, Neural Flow, and tPatchGNN. Dataset-family averages are computed over the processed series in each family. These tables use the dataset headers defined in section D.1. Each event-sequence interface configuration specifies a frozen TSFM and an interval encoding that uses either raw or log-transformed intervals. The baseline methods are evaluated in two setups: joint timing prediction and oracle time. Under joint timing prediction, the model predicts both the next value and the elapsed time until the next event. Its interval-prediction head is trained with raw or log-transformed interval targets. Under oracle time, the ground-truth next-event timestamp is supplied as the query time for the value decoder. Because oracle time does not predict timing, it reports value-at-given-time error only. Table 18 reports normalized value MAE and relative timing MAE, and table 19 reports normalized value MSE and relative timing MSE.

Table 18: Complete Time-IMM MAE results for frozen TSFMs using the event-sequence interface and trained irregular baselines. Value is normalized value error; Time is relative timing error and is omitted for oracle time. Lower is better.

Model	Interval	CES		Cluster		EPA		FNSPID		GDELT		ILI		MIMIC		Repo		Student	
		Value	Time	Value	Time	Value	Time	Value	Time	Value	Time	Value	Time	Value	Time	Value	Time	Value	Time
Chronos-2	Raw	0.696	0.501	0.326	8.039	0.120	0.020	0.092	0.146	0.611	1.440	0.197	0.008	0.400	1.520	0.258	0.692	0.571	0.004
	Log	0.696	0.508	0.328	7.932	0.118	0.022	0.092	0.149	0.611	1.434	0.196	0.010	0.399	1.515	0.257	0.711	0.571	0.004
CTSM 1.0	Raw	0.705	0.502	0.323	10.22	0.121	0.019	0.094	0.227	0.646	1.599	0.205	0.009	0.371	1.513	0.267	0.726	0.601	0.007
	Log	0.705	0.530	0.323	7.959	0.121	0.021	0.094	0.227	0.646	1.430	0.205	0.023	0.371	1.533	0.267	0.685	0.601	0.005
TimesFM 2.5	Raw	0.696	0.507	0.329	8.149	0.121	0.017	0.092	0.378	0.615	1.441	0.195	0.008	0.373	1.508	0.262	0.678	0.587	0.005
	Log	0.696	0.507	0.329	7.977	0.121	0.021	0.092	0.383	0.615	1.432	0.195	0.012	0.373	1.532	0.262	0.688	0.587	0.004
Toto 1.0	Raw	0.706	0.505	0.345	8.194	0.117	0.016	0.091	0.236	0.618	1.480	0.195	0.010	0.372	1.582	0.264	0.656	0.601	0.005
	Log	0.706	0.515	0.345	8.052	0.117	0.017	0.091	0.218	0.618	1.471	0.195	0.018	0.372	1.522	0.264	0.685	0.601	0.005
Toto 2.0	Raw	0.684	0.493	0.298	7.947	0.113	0.020	0.091	0.081	0.609	1.429	0.174	0.008	0.360	1.448	0.251	0.578	0.594	0.005
	Log	0.684	0.501	0.298	7.914	0.113	0.023	0.091	0.099	0.609	1.428	0.174	0.008	0.360	1.472	0.251	0.594	0.594	0.012
CRU	Raw	0.769	0.538	0.917	9.531	1.408	0.102	1.641	1.325	0.657	1.664	1.294	0.991	0.892	2.054	1.319	2.003	0.786	0.975
	Log	0.769	0.822	0.628	8.050	1.412	0.783	1.636	1.411	0.658	1.746	1.293	0.991	0.891	2.132	1.318	2.024	0.786	0.982
	Oracle	0.722	–	0.509	–	0.321	–	0.720	–	0.659	–	0.720	–	0.867	–	0.806	–	0.711	–
Latent ODE	Raw	0.768	0.999	0.917	9.092	0.946	0.997	1.529	1.444	0.658	1.822	1.234	0.992	0.891	2.171	1.239	2.048	0.793	1.003
	Log	0.769	1.040	0.519	8.099	0.349	1.069	0.832	1.445	0.657	1.852	0.911	0.992	0.893	2.297	0.959	2.048	0.777	1.003
	Oracle	0.719	–	0.470	–	0.348	–	0.768	–	0.659	–	0.757	–	0.839	–	0.808	–	0.694	–
Neural Flow	Raw	0.769	0.568	0.866	11.94	1.431	0.102	1.640	0.643	0.660	1.639	1.289	0.981	0.894	2.143	1.317	1.833	0.796	0.541
	Log	0.766	1.006	0.502	8.189	0.402	0.926	0.628	1.446	0.661	1.754	0.689	0.992	0.884	2.350	0.814	2.049	0.757	1.004
	Oracle	0.712	–	0.473	–	0.347	–	0.815	–	0.658	–	0.735	–	0.848	–	0.787	–	0.707	–
tPatchGNN	Raw	0.774	0.646	0.869	10.10	1.371	1.040	1.474	0.891	0.661	1.638	1.254	0.964	0.895	2.286	1.325	1.890	0.782	0.748
	Log	0.762	1.066	0.457	8.183	0.347	0.933	0.552	1.446	0.663	1.808	0.429	0.992	0.622	2.297	0.543	2.049	0.662	1.004
	Oracle	0.751	–	0.483	–	0.431	–	0.720	–	0.657	–	0.500	–	0.622	–	0.642	–	0.652	–

Table 19: Complete Time-IMM MSE results for frozen TSFMs using the event-sequence interface and trained irregular baselines. Value is normalized value error; Time is relative timing error and is omitted for oracle time. Lower is better.

Model	Interval	CES		Cluster		EPA		FNSPID		GDELT		ILI		MIMIC		Repo		Student	
		Value	Time	Value	Time	Value	Time	Value	Time	Value	Time	Value	Time	Value	Time	Value	Time	Value	Time
Chronos-2	Raw	0.837	0.421	0.199	957.6	0.092	0.555	0.062	0.174	1.049	6.097	0.139	0.008	0.715	74.19	0.437	3.890	0.732	0.020
	Log	0.835	0.426	0.197	945.9	0.091	0.648	0.062	0.187	1.048	5.922	0.138	0.008	0.713	72.30	0.434	4.155	0.732	0.020
CTSM 1.0	Raw	0.818	0.410	0.197	947.5	0.106	0.597	0.061	0.206	1.005	5.347	0.151	0.008	0.721	69.80	0.416	3.263	0.754	0.023
	Log	0.818	0.494	0.197	957.0	0.106	0.654	0.061	0.253	1.005	6.076	0.151	0.010	0.721	80.94	0.416	3.449	0.754	0.020
TimesFM 2.5	Raw	0.831	0.428	0.195	934.9	0.107	0.435	0.062	0.485	1.033	5.795	0.152	0.008	0.734	69.88	0.437	3.554	0.751	0.020
	Log	0.831	0.430	0.195	956.0	0.107	0.614	0.062	0.591	1.033	5.946	0.152	0.008	0.734	78.01	0.437	4.070	0.751	0.020
Toto 1.0	Raw	0.876	0.431	0.204	944.9	0.102	0.568	0.064	0.283	1.110	6.871	0.157	0.008	0.767	90.99	0.451	3.908	0.810	0.024
	Log	0.876	0.425	0.204	946.2	0.102	0.599	0.064	0.256	1.109	5.652	0.157	0.015	0.767	76.60	0.451	4.283	0.810	0.024
Toto 2.0	Raw	0.811	0.410	0.229	940.8	0.092	0.571	0.061	0.139	1.045	5.907	0.128	0.008	0.776	68.15	0.430	3.537	0.789	0.020
	Log	0.811	0.421	0.229	936.7	0.092	0.635	0.061	0.157	1.045	5.885	0.128	0.008	0.776	71.79	0.430	3.746	0.789	0.020
CRU	Raw	0.926	0.472	1.545	900.5	2.466	1.632	3.303	2.528	1.014	7.848	3.264	0.990	1.649	151.3	2.523	11.66	1.037	0.969
	Log	0.926	1.081	0.729	974.2	2.466	2.231	3.284	2.764	1.014	8.510	3.260	0.989	1.643	152.4	2.521	11.74	1.037	0.984
	Oracle	0.844	–	0.418	–	0.282	–	0.876	–	1.013	–	1.558	–	1.599	–	1.290	–	0.897	–
Latent ODE	Raw	0.926	1.450	1.545	903.8	1.486	2.613	2.945	2.860	1.013	8.816	3.072	0.992	1.649	152.6	2.310	11.84	1.054	1.026
	Log	0.926	1.538	0.458	975.8	0.319	2.765	1.116	2.862	1.014	8.916	1.863	0.992	1.647	153.5	1.747	11.84	1.024	1.026
	Oracle	0.840	–	0.396	–	0.308	–	0.988	–	1.009	–	1.643	–	1.549	–	1.266	–	0.862	–
Neural Flow	Raw	0.926	0.534	1.380	928.6	2.529	1.632	3.300	0.904	1.015	6.431	3.246	0.970	1.650	152.0	2.520	10.89	1.061	0.365
	Log	0.921	1.460	0.454	982.2	0.369	2.517	0.714	2.864	1.015	8.483	1.244	0.992	1.629	153.5	1.392	11.84	0.981	1.027
	Oracle	0.825	–	0.372	–	0.305	–	1.097	–	1.013	–	1.587	–	1.568	–	1.223	–	0.891	–
tPatchGNN	Raw	0.932	0.693	1.377	896.6	2.341	2.678	2.707	1.536	1.014	7.581	3.043	0.937	1.642	152.5	2.616	11.39	1.030	0.646
	Log	0.915	1.593	0.304	981.7	0.263	2.482	0.560	2.865	1.014	8.729	0.739	0.992	1.023	153.3	0.708	11.84	0.812	1.027
	Oracle	0.897	–	0.377	–	0.348	–	0.883	–	1.007	–	0.946	–	1.062	–	0.876	–	0.802	–

F.3 Complete EasyTPP Timing and Event-Type Results

The following tables report the complete EasyTPP results grouped by metric. TSFM configurations cross raw and log-transformed interval encodings with one-hot and frequency-ordered scalar event-type encodings. TPP baselines report NHP, THP, AttNHP, and IFTPP using their native event-type models. Table 20 reports timing MAE, table 21 reports timing MSE, and table 22 reports event-type error.

Table 20: Complete EasyTPP native-scale timing MAE results. Type Repr. denotes the TSFM event-type representation; TPP baselines use native event-type models. Lower is better.

Model	Interval	Type Repr.	Dataset						
			Amazon	Earth.	Retweet	Stack.	Taobao	Taxi	Volcano
Chronos-2	Raw	One-hot	0.270	0.800	2448	0.696	0.047	0.164	1225
		Freq. rank	0.269	0.812	2434	0.704	0.048	0.166	1225
	Log	One-hot	0.273	0.804	2354	0.698	0.047	0.164	1403
		Freq. rank	0.272	0.807	2353	0.702	0.048	0.166	1403
CTSM 1.0	Raw	One-hot	0.294	0.820	2587	0.778	0.062	0.181	1566
		Freq. rank	0.294	0.820	2587	0.778	0.062	0.181	1566
	Log	One-hot	0.303	0.763	2331	0.705	0.059	0.172	953.1
		Freq. rank	0.303	0.763	2331	0.705	0.059	0.172	953.1
TimesFM 2.5	Raw	One-hot	0.262	0.774	2343	0.698	0.046	0.163	1187
		Freq. rank	0.262	0.774	2343	0.698	0.046	0.163	1187
	Log	One-hot	0.264	0.761	2326	0.687	0.045	0.162	992.7
		Freq. rank	0.264	0.761	2326	0.687	0.045	0.162	992.7
Toto 1.0	Raw	One-hot	0.301	0.866	2426	0.752	0.047	0.176	1238
		Freq. rank	0.293	0.865	2427	0.730	0.046	0.175	1216
	Log	One-hot	0.305	0.888	2352	0.764	0.047	0.176	1603
		Freq. rank	0.297	0.888	2353	0.736	0.046	0.176	1473
Toto 2.0	Raw	One-hot	0.295	0.801	2370	0.706	0.045	0.167	1169
		Freq. rank	0.285	0.816	2384	0.717	0.047	0.168	1103
	Log	One-hot	0.297	0.802	2333	0.702	0.045	0.166	1051
		Freq. rank	0.288	0.819	2396	0.732	0.047	0.167	1100
NHP	–	-	0.335	0.947	3073	0.931	0.075	0.221	1139
THP	–	-	0.375	1.065	2569	0.995	0.060	0.211	979.7
AttNHP	–	-	0.540	0.885	4804	0.785	0.046	0.226	1171
IFTPP	–	-	0.300	0.976	2289	0.934	0.042	0.205	886.5

Table 21: Complete EasyTPP native-scale timing MSE results. Type Repr. denotes the TSFM event-type representation; TPP baselines use native event-type models. Lower is better.

Model	Interval	Type Repr.	Dataset						
			Amazon	Earth.	Retweet	Stack.	Taobao	Taxi	Volcano
Chronos-2	Raw	One-hot	0.154	2.045	2.70×10^8	1.373	0.019	0.112	2.24×10^7
		Freq. rank	0.156	2.090	2.64×10^8	1.391	0.020	0.113	2.24×10^7
	Log	One-hot	0.156	2.163	2.53×10^8	1.417	0.019	0.117	6.89×10^7
		Freq. rank	0.158	2.165	2.55×10^8	1.396	0.020	0.117	6.89×10^7
CTSM 1.0	Raw	One-hot	0.137	1.957	2.57×10^8	1.374	0.019	0.112	2.19×10^7
		Freq. rank	0.137	1.957	2.57×10^8	1.374	0.019	0.112	2.19×10^7
	Log	One-hot	0.140	1.808	2.49×10^8	1.248	0.019	0.103	1.70×10^7
		Freq. rank	0.140	1.808	2.49×10^8	1.248	0.019	0.103	1.70×10^7
TimesFM 2.5	Raw	One-hot	0.149	1.870	2.48×10^8	1.320	0.017	0.100	2.07×10^7
		Freq. rank	0.149	1.870	2.48×10^8	1.320	0.017	0.100	2.07×10^7
	Log	One-hot	0.150	1.854	2.49×10^8	1.279	0.017	0.099	1.63×10^7
		Freq. rank	0.150	1.854	2.49×10^8	1.279	0.017	0.099	1.63×10^7
Toto 1.0	Raw	One-hot	0.203	2.340	2.60×10^8	1.644	0.019	0.114	2.66×10^7
		Freq. rank	0.197	2.334	2.60×10^8	1.556	0.019	0.114	2.60×10^7
	Log	One-hot	0.205	2.454	2.47×10^8	1.673	0.019	0.115	7.37×10^7
		Freq. rank	0.199	2.453	2.45×10^8	1.559	0.019	0.115	4.73×10^7
Toto 2.0	Raw	One-hot	0.193	2.092	2.50×10^8	1.412	0.018	0.105	2.27×10^7
		Freq. rank	0.181	2.132	2.52×10^8	1.449	0.018	0.105	2.03×10^7
	Log	One-hot	0.193	2.170	2.47×10^8	1.412	0.018	0.105	1.75×10^7
		Freq. rank	0.184	2.531	1.07×10^9	10.33	0.018	0.105	2.72×10^7
NHP	–	-	0.235	2.636	2.73×10^8	2.123	0.050	0.178	1.59×10^7
THP	–	-	0.213	2.935	2.77×10^8	2.413	0.020	0.141	1.65×10^7
AttNHP	–	-	1.498	2.490	3.04×10^8	1.714	0.017	0.139	1.65×10^7
IFTPP	–	-	0.210	4.177	2.47×10^8	2.670	0.017	0.159	1.50×10^7

Table 22: Complete EasyTPP type error results. Type Repr. denotes the TSFM event-type representation; TPP baselines use native event-type models. Lower is better. Volcano has one event type, so its type error is 0 for all methods and should not be read as a categorical prediction comparison.

Model	Interval	Type Repr.	Dataset						
			Amazon	Earth.	Retweet	Stack.	Taobao	Taxi	Volcano
Chronos-2	Raw	One-hot	0.710	0.636	0.412	0.604	0.397	0.213	0.000
		Freq. rank	0.784	0.664	0.408	0.660	0.521	0.380	0.000
	Log	One-hot	0.710	0.636	0.411	0.604	0.397	0.213	0.000
		Freq. rank	0.783	0.664	0.409	0.660	0.521	0.379	0.000
CTSM 1.0	Raw	One-hot	0.735	0.657	0.424	0.643	0.405	0.211	0.000
		Freq. rank	0.877	0.714	0.426	0.806	0.694	0.258	0.000
	Log	One-hot	0.735	0.657	0.424	0.643	0.405	0.211	0.000
		Freq. rank	0.877	0.714	0.426	0.806	0.694	0.258	0.000
TimesFM 2.5	Raw	One-hot	0.713	0.625	0.406	0.603	0.396	0.212	0.000
		Freq. rank	0.805	0.675	0.413	0.679	0.542	0.327	0.000
	Log	One-hot	0.713	0.625	0.406	0.603	0.396	0.212	0.000
		Freq. rank	0.805	0.675	0.413	0.679	0.542	0.327	0.000
Toto 1.0	Raw	One-hot	0.743	0.660	0.427	0.667	0.403	0.360	0.000
		Freq. rank	0.722	0.620	0.405	0.611	0.445	0.408	0.000
	Log	One-hot	0.744	0.660	0.427	0.667	0.403	0.359	0.000
		Freq. rank	0.722	0.621	0.405	0.613	0.445	0.408	0.000
Toto 2.0	Raw	One-hot	0.720	0.660	0.418	0.619	0.409	0.204	0.000
		Freq. rank	0.770	0.626	0.417	0.652	0.472	0.256	0.000
	Log	One-hot	0.720	0.659	0.417	0.619	0.409	0.205	0.000
		Freq. rank	0.770	0.630	0.415	0.656	0.472	0.256	0.000
NHP	–	-	0.687	0.528	0.391	0.554	0.398	0.105	0.000
THP	–	-	0.661	0.528	0.509	0.550	0.402	0.092	0.000
AttNHP	–	-	0.743	0.537	0.407	0.626	0.393	0.555	0.000
IFTPP	–	-	0.644	0.528	0.399	0.542	0.387	0.082	0.000

G LIMITATIONS

Forecast horizon. This paper evaluates one-step next-event forecasting. The results show that frozen TSFMs can forecast the value and timing of the next observed event through the event-sequence interface. This finding establishes a foundation for multi-event extensions. Multi-event forecasting introduces recursive rollout and choices about aligning forecasts to future events. Evaluating it therefore requires a separate protocol. We leave this evaluation to future work.

Output interpretation. The reported TSFM value and timing results evaluate point predictions rather than predictive distributions. Accordingly, our conclusions concern point accuracy for next-event values and timestamps, together with decoded event-type performance on EasyTPP. These point-forecast results provide a foundation for future evaluation of the quantile and distributional forecasts available from these TSFMs. Future work can evaluate the quality of quantile and distributional forecasts available from the evaluated TSFMs.

Benchmark overlap. We also do not audit the full pretraining corpora of all evaluated TSFMs for benchmark overlap. The primary Time-IMM comparison uses a paired within-checkpoint design: both interfaces are evaluated with the same frozen TSFM on the same held-out examples. The resulting performance difference therefore provides direct within-checkpoint evidence that the event-sequence interface outperforms regular-grid interfaces on the reported task.