CONTEXT IN TIME SERIES: BENCHMARKING AND ALIGNING TIME-SERIES LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Existing benchmarks for time-series language models can often be solved using either text or time-series alone, making it difficult to assess genuine cross-modal reasoning. In this paper, we introduce the Context in Time-Series Benchmark (CiT-Bench), which comprises forecasting, imputation, and anomaly detection tasks designed such that textual context is necessary for accurate prediction. The text states a future event or constraint, describes an event inside a missing block, or singles out the true anomaly among distractors. To facilitate model development on CiT-Bench, we additionally construct a language–time-series alignment dataset for training. Using this dataset, we show that improved understanding of both textual and time-series inputs can be achieved by training only the embeddings of a small vocabulary of time-series words, without fully fine-tuning the language model. We provide a unified framework for evaluating and improving cross-modal reasoning between language and time-series. Our benchmark is available at https://huggingface.co/anonymized.

1 INTRODUCTION

Time-series are a ubiquitous data modality across real-world domains and systems, including finance (Zhu et al., 2025; Kim et al., 2026), healthcare (Ghassemi et al., 2015), energy (Goudarzi et al., 2019), and industrial monitoring (Gryllias & Antoniadis, 2012). Modeling temporal observations is therefore fundamental to a wide range of downstream applications. This has motivated extensive research on core time-series tasks, such as forecasting, imputation, and anomaly detection. Machine learning and deep learning approaches typically address these tasks by designing specialized architectures (Zhou et al., 2021; Wu et al., 2021; Kim et al., 2024), input representations (Nie et al., 2023; Wu et al., 2023), and training objectives (Yue et al., 2022; Woo et al., 2022) that account for the temporal structure and statistical characteristics of time-series data.

Large language models (LLMs) extend conventional time-series models by incorporating textual context. For example, forecasting often depends on information beyond the observed values, such as historical constraints or upcoming events. This has motivated time-series language models that integrate time-series representations with LLMs (Xie et al., 2025; Wang et al., 2025a;b; Lin et al., 2026).

Evaluating whether these models can use textual context requires datasets in which the text is necessary for prediction, yet such datasets are difficult to construct because paired series and text are scarce. Context is Key (CiK) (Williams et al., 2025) addresses this gap by pairing each series with textual context required for forecasting. However, we observe that, existing time-series language models achieve lower performance than recent general-purpose LLMs that receive time-series as decimal text (Figure 1). These results suggest that time-series training does not necessarily improve the use of textual context and may instead degrade it, although CiK evaluates only forecasting performance.

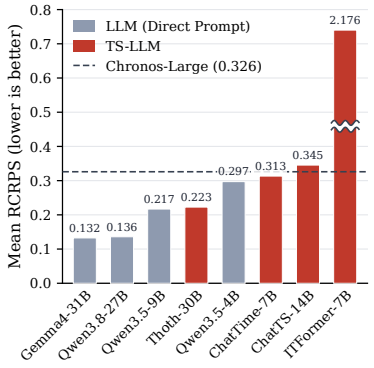

Figure 1: **Mean RCRPS on Context is Key.** Recent general-purpose LLMs (gray) outperform time-series language models (red).

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.

In this paper, motivated by CiK, we introduce the Context-in Time-Series Benchmark (CiT-Bench), which extends context-dependent evaluation beyond forecasting to a broader set of fundamental time-series tasks. CiT-Bench pairs time-series with textual context and constructs tasks that cannot be reliably solved from the time-series alone. Specifically, we consider forecasting, imputation, and anomaly detection, with textual context serving a distinct role in each setting. It states a future event or constraint that the observed history does not reveal, describes events occurring within unobserved segments, or specifies attributes that distinguish true anomalies from plausible distractors. By construction, CiT-Bench directly evaluates whether a model can integrate linguistic context with temporal observations rather than relying predominantly on either modality in isolation.

Beyond evaluation, we construct CiT-Corpus, a language–time-series alignment dataset of diverse tasks built from series that are disjoint from CiT-Bench. To train on it, we consider existing fine-tuning approaches and, since fine-tuning on time-series cost the use of text (Figure 1), introduce NEOTS, which keeps the LLM frozen and adds a small vocabulary of time-series words whose embeddings are the only trained parameters. Training on CiT-Corpus improves joint reasoning over series and text on CiT-Bench, and together the benchmark, the corpus, and the method form a framework for evaluating and improving cross-modal reasoning between language and time-series.

2 RELATED WORK

2.1 TIME-SERIES LANGUAGE MODELS

Time-series language models attach numerical series to pretrained language models to enable natural-language description, reasoning, or forecasting. LLMTime (Gruver et al., 2023) represents series as digit strings without training; ChatTS (Xie et al., 2025) and ITFormer (Wang et al., 2025b) use continuous encoders; Thoth (Lin et al., 2026) writes values as text; and ChatTime (Wang et al., 2025a) and TempoGPT (Zhang et al., 2026) introduce discrete series tokens. Except for ITFormer, which trains only a bridging module, these methods update the backbone through fine-tuning or low-rank adaptation, with TempoGPT first training the new embeddings alone. Their text is written about the paired series, and evaluation focuses on series capabilities rather than how training affects textual knowledge already present in the backbone.

Adding a modality through the input embeddings of a frozen language model has several precedents. Prompt tuning learns continuous input vectors (Lester et al., 2021), while BLIP-2 trains a module that maps visual inputs into the embedding sequence of a frozen language model (Li et al., 2023). Time-LLM (Jin et al., 2024) similarly maps time-series patches into a frozen backbone but predicts through a linear head rather than text. Neologism learning (Hewitt et al., 2026) trains only the input embeddings of a few new words while keeping the backbone and output vocabulary fixed, but has been applied to textual concepts rather than, to our knowledge, numerical modalities.

2.2 TIME-SERIES BENCHMARKS

Time-series language model benchmarks typically use question-answering tasks in which text specifies the query and the series determines the answer. TimeSeriesExam (Cai et al., 2024) asks multiple-choice questions about synthetic-series properties, MMTS-Bench (Yin et al., 2026) evaluates capabilities from structural awareness to temporal reasoning and cross-modal alignment, and QuAnTS (Divo et al., 2025) includes binary, multiple-choice, and open-ended questions on synthetic human-motion trajectories.

Several benchmarks also provide contextual text. Time-MQA (Kong et al., 2025) includes background descriptions, ARFBench (Xie et al., 2026) pairs telemetry with metric descriptions, and TSR-Suite (Guan et al., 2026) emphasizes context sufficiency across multiple reasoning and forecasting tasks. However, sufficiency does not imply necessity, since tasks may remain solvable without the text. Context is Key (Williams et al., 2025) enforces necessity by pairing numerical histories with text required for accurate forecasting and evaluates predictions with a modified continuous ranked probability score that emphasizes regions of interest and constraint violations. To our knowledge, our benchmark is the first dataset that constructs imputation or anomaly detection tasks with necessary textual context.

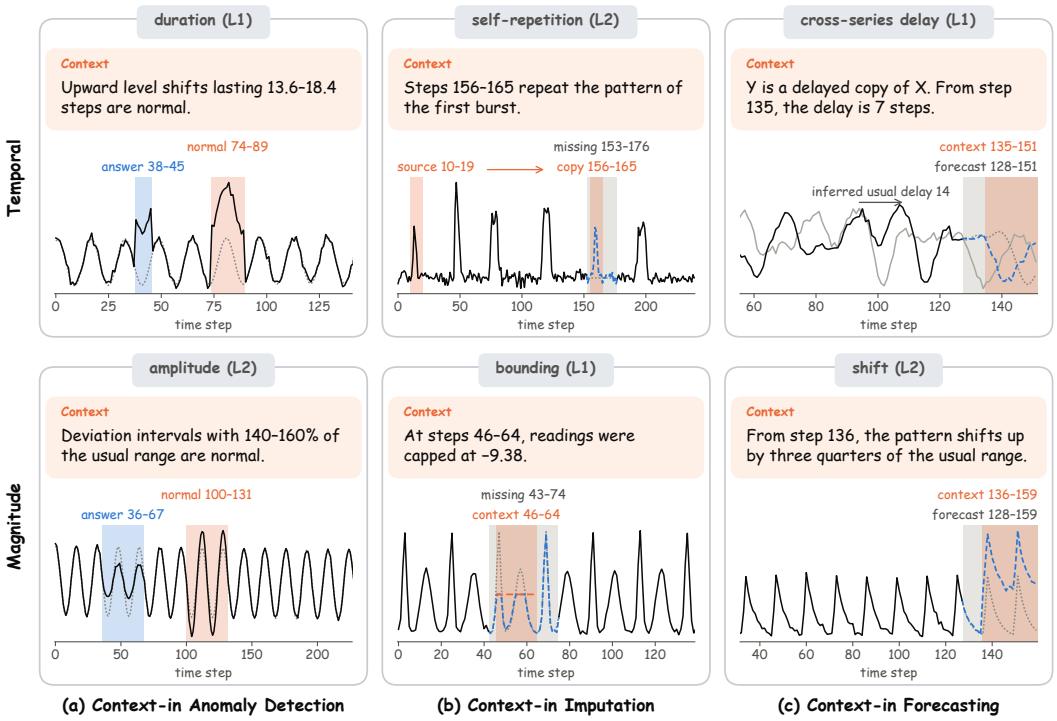

Figure 2: **Examples of CiT-Bench.** Columns show the three tasks, and rows show the category of event. Blue marks targets, and orange highlights event-related intervals and bounds. Gray shading denotes missing or forecast windows, while dotted gray lines show the usual signal.

3 Context-in Time-Series Benchmark

3.1 Benchmark Construction

To enable comprehensive evaluation of time-series reasoning with context, we consider three main tasks in time-series domain: **context-in anomaly detection, context-in imputation**, and **context-in forecasting**. As shown in Figure 2, each instance consists of an observed series, textual context, and a target. For imputation and forecasting, the target spans the missing block or forecast horizon and contains the subsequence described by the context, which we call the region of interest (RoI). The RoI is therefore absent from the observed series, while the target outside the RoI follows the pattern established by the observations. For anomaly detection, the observed series contains two events of the same type that deviate from the usual pattern but differ in one attribute, such as shift magnitude. The context identifies one event as normal operation, and the target is the interval corresponding to the other event.

Across all three tasks, the context specifies an event attribute, and the target is determined by the corresponding event parameters. Because this construction requires exact control of these parameters, all series are synthetic. Each series consists of a periodic or aperiodic base signal with additive noise and a controlled event, such as a level shift, a change in amplitude or repetition interval, a repetition of an earlier burst, or a change in the delay between two series. The base signals, event types, and parameter ranges are detailed in Appendix A.

We further organize each task along two independent axes, **category** and **level**, which capture what event attribute the context specifies and how it is specified, respectively (Table 1). Categories are either *magnitude*, concerning signal values, or *temporal*, concerning temporal structure. Levels vary the form of the context while keeping the observations and ground truth fixed: Level 1 specifies the attribute directly with numerical values, whereas Level 2 expresses it relative to a reference quantity that must be estimated from the observed time series. Each task contains 100 queries for each of the four category–level combinations, and we report results along both axes.

Table 1: **Category and level in CiT-Bench.** Category indicates what aspect of the event attribute is specified, while level indicates how the attribute is expressed in the context.

Axis	Value	The context specifies	Example
Category	Magnitude	level, amplitude, bound, or trend rate	readings capped at -9.38
	Temporal	interval, duration, repeat, or delay	steps 156–165 repeat the burst
Level	1	the attribute as a number	a shift of 17.09 units
	2	the attribute relative to the usual pattern	half the usual range

Table 2: **Summary of CiT-Bench.** Paired queries share the same scene.

Task	Scenes	Paired	Queries	Output	Metric
Anomaly detection	100	Y	400	interval (s, e)	accuracy, consistency
Imputation	200	N	400	missing block	RCRPS
Forecasting	200	N	400	continuation	RCRPS

We next describe each task in detail. Let us assume that each query consists of a textual context C and a time series $x_{1:T}$ of T time steps, where $x_{a:b}$ denotes the values from steps a through b. Across all three tasks, the observations alone are insufficient to determine the target, and C provides the additional information required.

Context-in Anomaly Detection. The series x contains two unmarked deviations of the same type, separated by at least one time step. The context C specifies benign and anomalous conditions for one attribute, such as duration or amplitude, under which it is considered normal. Exactly one interval violates this condition, and the task is to predict this interval (s, e):

$$f(x_{1:T}; C) = (\hat{s}, \hat{e}) \tag{1}$$

where f denotes a model under evaluation. To verify that the prediction depends on C, we construct paired queries with the same series but different contexts such that different deviations are anomalous (Paired column in Table 2). Therefore, a model that correctly uses the context should identify the appropriate target under both contexts in each pair.

Context-in Imputation. A block $x_{t+1:t+r}$ of r consecutive steps is missing. C describes an event within the region of interest (s, e) with $t + 1 < s < e < t + r$. The event changes a magnitude or temporal attribute of the series. For example by halving the usual range or doubling the usual period, while values outside (s, e) follow the usual pattern. Therefore, the task is to fill the block:

$$f\big(x_{1:t}, x_{t+r+1:T}; C\big) = \hat{x}_{t+1:t+r}, \tag{2}$$

Furthermore, we consider imputation settings in which information from another time series is required. For example, in industrial monitoring, a downstream sensor may normally respond to an upstream sensor after a characteristic delay. This usual delay is inferred from the observed portions of the two series, while C specifies the delay within (s, e) directly at Level 1 or relative to the usual delay at Level 2. The reference series is fully observed, and only the target series is imputed.

Context-in Forecasting. The task is to forecast the H steps that follow $x_{1:T}$. C describes an event that changes a magnitude or temporal attribute of the series and states the step s at which it begins. The event lasts until the end of the horizon, so the region of interest is $(s, T + H)$, where $T + 1 < s < T + H$ and the values before step s follow the usual pattern. Therefore, the task is to forecast the block:

$$f\big(x_{1:T}; C\big) = \hat{x}_{T+1:T+H}, \tag{3}$$

As in imputation, we also consider forecasting settings that require information from another time series. For example, in traffic monitoring, an upstream traffic signal may influence a downstream signal after a characteristic delay. The reference series is observed throughout the forecast horizon, while C specifies how this delay changes from step s onward.

Note that **in all three tasks, the target cannot be correctly predicted without understanding** C, ensuring that performance reflects the ability to integrate textual context with time-series data.

Table 3: **Evaluated models.** Parameter counts include only the weights used in our experiments, excluding unused vision towers and multi-token prediction heads. Thoth-30B is a mixture-of-experts model with approximately 3.3B active parameters per token.

Model	Checkpoint or API version	Params	Series input	Inference
Open-weight LLMs				
Gemma4-31B	`google/gemma-4-31B-it`	30.7B	text	vLLM 0.28.0
Qwen3.8-27B	`Qwen/Qwen3.8-27B`	26.9B	text	vLLM 0.27.1
Qwen3.5-9B	`Qwen/Qwen3.5-9B`	9.0B	text	vLLM 0.27.1
Time series LLMs				
Thoth-30B	`thuml/Thoth-30B-A3B`	30.5B	text	vLLM 0.27.1
ChatTS-14B	`bytedance-research/ChatTS-14B`	14.9B	encoder	Transformers 5.4.0
ITFormer-7B	`pandalin98/ITFormer-7B` on `Qwen/Qwen2.5-7B-Instruct`	7.7B	encoder	Transformers 4.52.4 for anomaly detection, vLLM 0.27.1 otherwise
ChatTime-7B	`ChengsenWang/ChatTime-1-7B-Chat`	6.8B	value tokens	vLLM 0.27.1
Closed models				
GPT-4o mini	`gpt-4o-mini-2024-07-18`	not public	text	OpenAI Chat Completions
Gemini 3.1 Flash-Lite	`gemini-3.1-flash-lite`	not public	text	Gemini API
Claude Haiku 4.5	`claude-haiku-4-5-20251001`	not public	text	Anthropic Messages API

3.2 Evaluation and Results

Evaluation. Based on CiT-Bench, we evaluate three model families: general-purpose open-weight LLMs (Open), time-series language models (Time-series), and closed-weight models (Closed). All models receive the series and context in a single answer-only prompt and return text outputs: one interval for anomaly detection and 25 sampled sequences for imputation and forecasting. Table 3 summarizes models that are evaluated. Prompt templates, decoding settings, output validation, and resampling rules are provided in Appendix B.4.

For anomaly detection, following (Gao et al., 2017), we report accuracy, where a predicted interval is correct if its intersection over union with the target is at least 0.5. We additionally report consistency, defined as the fraction of paired queries for which the model correctly identifies the target under both contexts; higher is better. For imputation and forecasting, we report the region-of-interest CRPS (RCRPS) from Context is Key (Williams et al., 2025), which evaluates the predictive distribution by separately weighting steps inside and outside $\mathcal{R}$; lower is better. We also report coverage, the fraction of queries for which the required number of valid samples is obtained, since RCRPS values computed from different numbers of samples are not directly comparable. Following Williams et al. (2025), we assign a fixed penalty score of 1 to instances whose model outputs cannot be parsed due to violations of the required output format even after resampling in allowed budgets.

Results. Table 4 summarizes the evaluation results. CiT-Bench remains challenging even for strong general-purpose LLMs. In anomaly detection, the best model reaches only 55.5% accuracy, and no model exceeds 20% consistency, indicating that models rarely identify the correct event under both contexts of a paired query. Imputation and forecasting exhibit a similar gap: among models with complete coverage, Level 2 generally yields higher RCRPS than Level 1, showing that interpreting context relative to a quantity inferred from the observations is more difficult than using explicitly stated numerical values. Second, general-purpose LLMs consistently outperform existing time-series language models across the three tasks. Among the latter, Thoth-30B performs strongest overall, but still trails the larger general-purpose models. This gap is particularly pronounced in anomaly detection, where most time-series models achieve low accuracy and near-zero consistency. These results suggest that time-series training alone does not necessarily translate into effective integration of textual context with numerical observations.

Table 4: **Performance on CiT-Bench by task, level, and category.** Acc., Cons., and Cov. denote accuracy, consistency, and coverage (%), respectively; higher is better. Lower RCRPS is better. ChatTime-7B returned no response in the required format in imputation or forecasting, so its coverage is 0 and its RCRPS is shown as N/A.

		Model	Level 1				Level 2				Overall	
			Temporal		Magnitude		Temporal		Magnitude			
			Acc.	Cons.	Acc.	Cons.	Acc.	Cons.	Acc.	Cons.	Acc.	Cons.
Context-in Anomaly Detection	Open	Gemma4-31B	**49.0**	**12.0**	**73.0**	**50.0**	**50.0**	**8.0**	**50.0**	**10.0**	**55.50**	**20.0**
		Qwen3.8-27B	32.0	6.0	50.0	18.0	28.0	0.0	39.0	6.0	37.25	7.5
		Qwen3.5-9B	18.0	4.0	25.0	4.0	21.0	0.0	26.0	2.0	22.50	2.5
	Time-series	Thoth-30B	16.0	0.0	**31.0**	**2.0**	20.0	0.0	**26.0**	0.0	**23.25**	0.5
		ChatTS-14B	**18.0**	0.0	21.0	**2.0**	**24.0**	**2.0**	16.0	0.0	19.75	**1.0**
		ITFormer-7B	1.0	0.0	2.0	0.0	1.0	0.0	3.0	0.0	1.75	0.0
		ChatTime-7B	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.00	0.0
	Closed	GPT-4o mini	15.0	2.0	16.0	2.0	13.0	0.0	14.0	0.0	14.50	1.0
		Gemini 3.1 Flash-Lite	**37.0**	**4.0**	**57.0**	**34.0**	**37.0**	2.0	37.0	**4.0**	**42.00**	**11.0**
		Claude Haiku 4.5	**37.0**	**4.0**	52.0	22.0	30.0	**4.0**	**38.0**	2.0	39.25	8.0
			Cov.	RCRPS	Cov.	RCRPS	Cov.	RCRPS	Cov.	RCRPS	Cov.	RCRPS
Context-in Imputation	Open	Gemma4-31B	100	**0.092**	100	0.131	100	0.141	100	0.156	100	0.130
		Qwen3.8-27B	100	0.101	100	**0.115**	100	**0.122**	100	**0.136**	100	**0.119**
		Qwen3.5-9B	60	0.500	55	0.534	57	0.523	55	0.546	57	0.526
	Time-series	Thoth-30B	54	**0.530**	71	**0.390**	56	**0.512**	68	**0.434**	62	**0.466**
		ChatTS-14B	54	0.553	43	0.643	42	0.653	29	0.766	42	0.654
		ITFormer-7B	92	1.618	95	0.641	94	1.553	98	1.547	95	1.340
		ChatTime-7B	0	N/A	0	N/A	0	N/A	0	N/A	0	N/A
	Closed	GPT-4o mini	96	0.190	97	0.204	97	0.181	94	0.242	96	0.204
		Gemini 3.1 Flash-Lite	100	**0.099**	100	0.125	100	**0.134**	100	0.143	100	**0.125**
		Claude Haiku 4.5	100	0.143	100	**0.124**	100	0.167	100	**0.142**	100	0.144
			Cov.	RCRPS	Cov.	RCRPS	Cov.	RCRPS	Cov.	RCRPS	Cov.	RCRPS
Context-in Forecasting	Open	Gemma4-31B	100	**0.074**	100	**0.076**	100	**0.104**	100	**0.107**	100	**0.090**
		Qwen3.8-27B	100	0.086	100	0.088	100	0.109	100	0.110	100	0.098
		Qwen3.5-9B	100	0.136	98	0.152	100	0.143	100	0.151	99.5	0.146
	Time-series	Thoth-30B	95	**0.167**	97	**0.160**	96	**0.159**	100	**0.152**	97	**0.159**
		ChatTS-14B	65	0.440	66	0.451	68	0.414	66	0.460	66	0.441
		ITFormer-7B	100	1.743	96	0.742	95	1.716	97	1.395	97	1.399
		ChatTime-7B	0	N/A	0	N/A	0	N/A	0	N/A	0	N/A
	Closed	GPT-4o mini	100	0.149	100	0.145	100	0.161	100	0.164	100	0.155
		Gemini 3.1 Flash-Lite	100	**0.075**	100	**0.060**	100	**0.116**	100	**0.087**	100	**0.084**
		Claude Haiku 4.5	100	0.130	100	0.098	100	0.150	100	0.126	100	0.126

Finally, coverage alone does not indicate successful time-series reasoning. ITFormer-7B, for example, produces valid outputs for most imputation and forecasting queries, yet obtains RCRPS above 1 overall, despite 1 being assigned to format failures. Its only substantially lower errors occur under Level 1 magnitude conditions, where the context directly provides the relevant numerical value. This behavior suggests that ITFormer struggles to recover the numerical scale from its encoded series and instead benefits from explicit numerical cues in the context. Its architecture may contribute to this failure mode: the frozen backbone receives the series only through a fixed set of encoder representations, while the prompt provides no direct indication of its numerical scale. At the other extreme, ChatTime-7B produces outputs in its own value-token representation, yielding no valid samples under our text-output protocol. Together, these results expose a distinction between producing syntactically valid time-series outputs and correctly grounding textual context in the numerical series.

Table 5: **Summary of CiT-Corpus.** Each task poses a question over one or more time series with a single exact answer.

Task	Question	Answer
Forecasting	Continue the series.	the next values
Imputation	Fill in the masked values, scattered or one contiguous gap.	the masked values
Counterfactual continuation	Continue the series as prescribed: constant, repeated, wave, or linear bridge.	the prescribed continuation
Constrained forecasting	Continue the series given one property of its future: bound, band, peak, or mean shift.	the true continuation
Captioning	Describe the pattern, noise, statistics, composite oscillation, or periodicity.	a sentence
Editing	Reverse the series in time, or remove its spikes.	the edited series
Arithmetic and read-out	Add, scale, negate, or combine series, or read a value at an index or the first index above a threshold.	the resulting series, a value, or an index
Multi-series relations	Compare several series: copy, split, lead/lag, locate a sub-sequence, match a continuation, or select by a property.	a label, an index, or a series
Forecasting with covariates	Continue a target series given a reference series with a known future or a lead.	the future of the target

4 CiT-Corpus for Language-Time Series Alignment

Our comprehensive evaluation on CiT-Bench shows that, consistent with Context is Key, current LLMs and time-series language models still struggle to integrate textual context with time-series observations. Notably, recent general-purpose LLMs given series as decimal text substantially outperform time-series language models, despite the latter being explicitly trained on time-series data. This suggests that general-purpose backbones already possess strong language capabilities that are not necessarily preserved by time-series fine-tuning. Therefore, in this section, to facilitate further progress on CiT-Bench, we construct a corpus that links language and time series to exact numerical answers. Additionally, we further explore an efficient post-training approach that learns a small vocabulary of time-series tokens on this corpus while keeping all backbone parameters fixed.

4.1 Training Data Construction

Large-scale time-series data paired with instructions and exact numerical answers are currently unavailable. To support further progress on CiT-Bench, we construct the **Context in Time-Series corpus (CiT-Corpus)**. It contains 31 tasks and 3.4M examples (Table 5), spanning four language–series relations: continuing a series under stated conditions, describing it, transforming it, and relating multiple series. Each example is a single-turn instruction over one or more series with a unique textual answer computed from the input and instruction. In continuation tasks, the same series is paired with different instructions and answers, ensuring that the target depends on the instruction rather than the history alone. CiT-Corpus fixes the numerical values, instruction, and answer while leaving the numerical input representation flexible. Context-dependent tasks from Section 3 are excluded.

The corpus is generated from two data sources. Continuation tasks use windows from the Chronos pre-training corpus (Ansari et al., 2024), which combines real time series with Gaussian-process samples but covers a relatively limited magnitude range. The remaining tasks use procedurally generated waveforms, enabling exact control over patterns, noise, and cross-series relations. These waveforms are rescaled to uniformly cover decimal orders from 10^{-1} to 10^{4}, extending supervision beyond the magnitude range of Chronos. Each example is generated by sampling a series, task, and task parameters, then computing the exact answer. Instructions are drawn from template pools with lexical slots (Honovich et al., 2023), with the final sentences of each pool held out from training. Numerical answers are expressed in the original units with precision determined by the largest absolute series value, maintaining comparable significant-digit precision across scales. Generation rules, parameter ranges, templates, corpus statistics, and the relation to CiT-Bench are detailed in Appendix G.

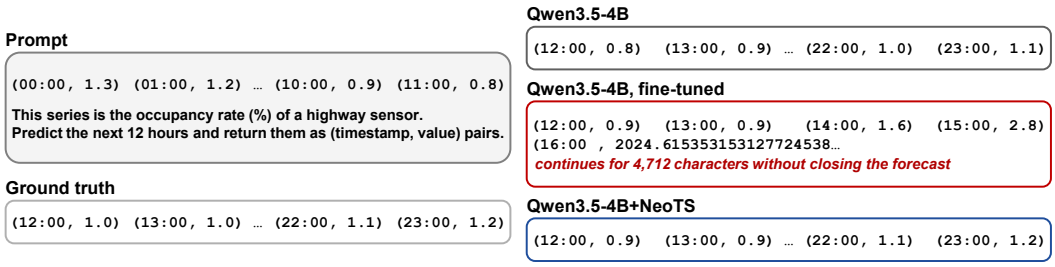

Figure 3: Sample query and corresponding answers from Qwen3.5-4B under three settings. Mean RCRPS over the benchmark (lower is better) is 0.297 for the frozen backbone and 0.267 with NeoTS; full fine-tuning cannot be scored because some outputs cannot be parsed as forecasts.

4.2 Learning Time-Series Words

The direct way to use CiT-Corpus is to fine-tune a language model on it, as the time-series language models are trained on their own data. Figure 3 shows the cost on Qwen3.5-4B, which is small enough for full fine-tuning: trained on CiT-Corpus as decimal text, it no longer closes its forecasts on Context is Key and cannot be scored, an instance of catastrophic forgetting (Luo et al., 2025). The usual remedies, mixing language data into the fine-tuning set (Ouyang et al., 2022), constraining the update (Kirkpatrick et al., 2017; Hu et al., 2022), or a further post-training stage (Wang et al., 2025a), add data and settings to be tuned for every backbone. We therefore leave the backbone untouched and change only what it reads, which preserves the answer format and improves the score with the same corpus.

The key idea behind NeoTS is to extend neologism learning (Hewitt et al., 2026) to numerical inputs. Although language models represent numbers as digit-string tokens, these tokens do not encode magnitude and precision as a unified semantic unit. As illustrated in Figure 4, NeoTS introduces a small vocabulary of time-series tokens while keeping the backbone frozen. Therefore, this design can be readily applied to existing LLMs.

A series of N values enters the prompt as a block of $N+3$ tokens between two delimiters: a scale word followed by N value words. The scale word encodes the power of ten that bounds the series, and each value word encodes one observation divided by that power, so that the values lie in $[-1, 1]$. Neither word is a discrete token. The embedding of a value word is a linear interpolation between the two neighboring rows of a trainable table with 201 grid points, so that the value enters without quantization error, and the embedding of the scale word is a continuous function of the exponent with two trainable vectors. A missing observation has its own row, and the two delimiters reuse the frozen embeddings of the image-boundary tokens of the backbone (Qwen Team, 2026). The embeddings are defined in Appendix D.

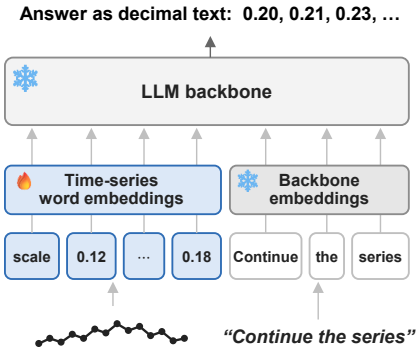

Figure 4: NeoTS. A series enters the prompt as a block of time-series words.

Every series in a prompt is rendered as a block, alone or together with its decimal list, and a transcription task from a block to its decimal list is added to the corpus (Appendix G). The vocabulary is trained for one epoch with the cross-entropy of the answer tokens, with all prompt tokens excluded from the loss. Since nothing bounds the norm of the new embeddings, we follow neologism learning and add a one-sided penalty that keeps each trainable vector $\mathbf{v}$, a row of the table or one of the two scale vectors, near the norm τ of a text embedding of the backbone,

$$\mathcal{L} = \mathcal{L}_{\mathrm{CE}} + \lambda \sum_{\mathbf{v}} \max\left(\|\mathbf{v}\|_2 - \tau,\ 0\right). \tag{4}$$

The training configuration is given in Appendix E, and the evaluation of Qwen3.8-27B+NeoTS on Context is Key in Appendix F.

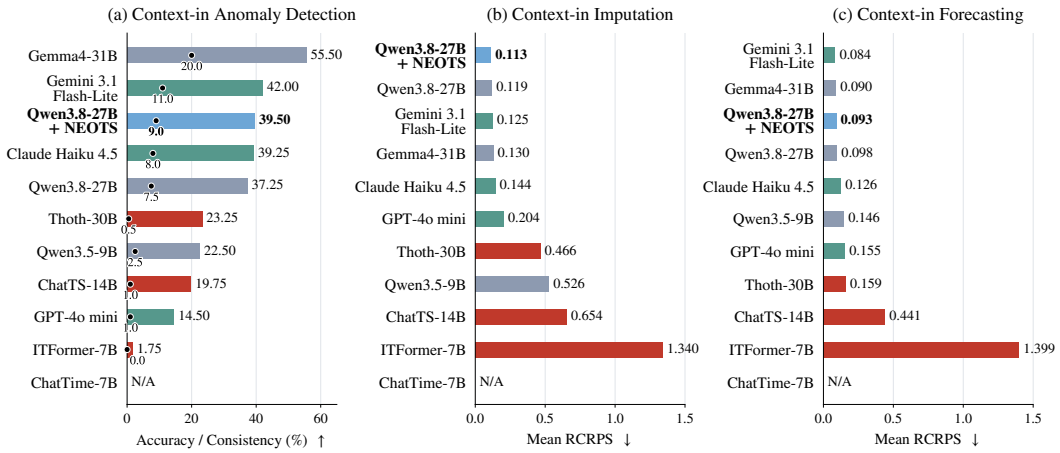

Figure 5: **Overall performance on CiT-Bench.** Bars show anomaly-detection accuracy and mean RCRPS for imputation and forecasting; dots show pair consistency. Blue bars denote Qwen3.8-27B + NEOTS. ChatTime-7B has 0% coverage and is therefore reported as N/A.

4.3 EVALUATION ON CIT-BENCH

We compare all models on the three CiT-Bench tasks, including Qwen3.8-27B with NEOTS, under the protocol of Section 3. As in training, the series is provided both through the learned time-series words and as decimal text, and answers are generated as text. Our primary comparison is against the frozen Qwen3.8-27B backbone, which receives the same decimal representation but without the learned time-series words. Figure 5 summarizes the overall results, with the remaining model results taken from Table 4. Adding NEOTS consistently improves the Qwen3.8-27B backbone across all three tasks. On anomaly detection, both accuracy and pair consistency increase, indicating that the gain is not explained solely by selecting a more salient event but also reflects improved use of the paired context. The improvement is also consistent on the generative tasks, reducing RCRPS for both imputation and forecasting. In particular, Qwen3.8-27B+NEOTS achieves the lowest imputation RCRPS among the evaluated models and remains competitive with the strongest general-purpose models on forecasting.

These results show that CiT-Corpus provides useful supervision for contextual time-series reasoning. Existing time-series language models still lag behind strong general-purpose LLMs on CiT-Bench, suggesting that time-series-only fine-tuning may improve numerical modeling without preserving broader language capabilities. Our results point to a complementary direction: keep the backbone frozen and learn a compact numerical interface that exposes time-series inputs to its existing language capabilities. This provides a parameter-efficient alternative to full-model adaptation for joint reasoning over language and time series.

5 CONCLUSION

Time series are ubiquitous in real-world systems, yet current LLMs and time-series language models still struggle to integrate numerical observations with textual context. We introduced CiT-Bench to evaluate this capability across context-in anomaly detection, imputation, and forecasting, where the target cannot be determined from the series alone. Our evaluation reveals substantial room for improvement in jointly understanding language and time series. A limitation of CiT-Bench is its reliance on synthetic data, as real-world time series are rarely accompanied by textual descriptions. Future work should therefore explore constructing a dataset using naturally occurring multimodal data. To support further progress, we introduced CiT-Corpus, a large-scale corpus that pairs time series and language with exact numerical answers, and suggested a method, an efficient post-training approach that learns only a small vocabulary of time-series tokens while keeping the language-model backbone frozen. Together, CiT-Bench, CiT-Corpus, and NEOTS provide a benchmark, training resource, and simple adaptation strategy for developing time-series language models that more effectively combine numerical signals with language.

AI USE STATEMENT

In this work, we used generative AI tools for implementing the benchmark generators, the validation and evaluation code and the model-running scripts, generating the synthetic benchmark data with that code, running and monitoring the model evaluations, and assisting with translation. We have not used generative AI tools for formulating mathematical claims, providing critical ingredients for proving mathematical claims, assisting in the writing of proofs, or supporting qualitative and thematic data analysis. Additionally, we used generative AI tools for drafting and editing manuscript text, including the appendices and tables, checking the text against the released code and data, and summarizing results for the authors to interpret. We have reviewed all AI-assisted work. LLM-generated code and writing was verified and tested for correctness by all authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

All series, contexts and targets of CiT-Bench are produced by code, and Appendix A gives the generation parameters, the event definitions, the balancing schedule, the context templates and the quality checks of the released sets. We will release an evaluation package that contains the three evaluation sets with their prompts, records and tags, the generator of the imputation and forecasting sets, the evaluation code with the parser, the seed rule, the retry schedule and the metrics, a fixed protocol file, the reference results of every reported model, and checksums with a verification script, so that the reported numbers can be recomputed from the released responses and a new model can be evaluated under the same protocol. Appendix B lists the checkpoints and API versions, the inference engines, the series input of each time series LLM, the complete prompts and the generation settings, including the seed rule and the cases in which a rerun does not reproduce every response. Appendix C defines the parsing rules, the failure score and the metrics. We will publicly release all source code on Hugging Face.

REFERENCES

Abdul Fatir Ansari, Lorenzo Stella, Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Rangapuram, Sebastian Pineda Arango, Shubham Kapoor, Jasper Zschiegner, Danielle Maddix Robinson, Hao Wang, Michael Mahoney, Kari Torkkola, Andrew Wilson, Michael Bohlke-Schneider, and Yuyang (Bernie) Wang. Chronos: Learning the language of time series. *Transactions of Machine Learning Research*, 2024. URL https://www.amazon.science/publications/chronos-learning-the-language-of-time-series.

Yifu Cai, Arjun Choudhry, Mononito Goswami, and Artur Dubrawski. Timeseriesexam: A time series understanding exam. In *NeurIPS Workshop on Time Series in the Age of Large Models*, 2024. URL https://openreview.net/forum?id=oXRX7ABgLz.

Felix Divo, Maurice Kraus, Anh Q Nguyen, Hao Xue, Imran Razzak, Flora D Salim, Kristian Kersting, and Devendra Singh Dhami. Quants: Question answering on time series. *arXiv preprint arXiv:2511.05124*, 2025.

Jiyang Gao, Chen Sun, Zhenheng Yang, and Ram Nevatia. Tall: Temporal activity localization via language query. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 5277–5285. IEEE, 2017.

Marzyeh Ghassemi, Marco Pimentel, Tristan Naumann, Thomas Brennan, David Clifton, Peter Szolovits, and Mengling Feng. A multivariate timeseries modeling approach to severity of illness assessment and forecasting in icu with sparse, heterogeneous clinical data. In *Proceedings of the AAAI conference on artificial intelligence*, volume 29, 2015.

Siavash Golkar, Mariel Pettee, Michael Eickenberg, Alberto Bietti, Miles Cranmer, Geraud Krawezik, Francois Lanusse, Michael McCabe, Ruben Ohana, Liam Holden Parker, et al. xval: A continuous number encoding for large language models. In *NeurIPS 2023 AI for Science Workshop*, 2023.

Shidrokh Goudarzi, Mohammad Hossein Anisi, Nazri Kama, Faiyaz Doctor, Seyed Ahmad Soleymani, and Arun Kumar Sangaiah. Predictive modelling of building energy consumption based on a hybrid nature-inspired optimization algorithm. *Energy and Buildings*, 196:83–93, 2019.

Nate Gruver, Marc Finzi, Shikai Qiu, and Andrew G Wilson. Large language models are zero-shot time series forecasters. *Advances in neural information processing systems*, 36:19622–19635, 2023.

Konstantinos C Gryllias and Ioannis A Antoniadis. A support vector machine approach based on physical model training for rolling element bearing fault detection in industrial environments. *Engineering Applications of Artificial Intelligence*, 25(2):326–344, 2012.

Tong Guan, Zijie Meng, Dianqi Li, Shiyu Wang, Chao-Han Huck Yang, Qingsong Wen, Zuozhu Liu, Sabato Siniscalchi, Ming Jin, and Shirui Pan. Timeomni-1: Incentivizing complex reasoning with time series in large language models. In *International Conference on Learning Representations*, volume 2026, pp. 152139–152170, 2026.

John Hewitt, Oyvind Tafjord, Robert Geirhos, and Been Kim. Neologism learning for controllability and self-verbalization. In *International Conference on Learning Representations*, volume 2026, pp. 89344–89371, 2026.

Or Honovich, Thomas Scialom, Omer Levy, and Timo Schick. Unnatural instructions: Tuning language models with (almost) no human labor. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 14409–14428, 2023.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

Ming Jin, Shiyu Wang, Lintao Ma, Zhixuan Chu, James Zhang, Xiaoming Shi, Pin-Yu Chen, Yuxuan Liang, Yuan-Fang Li, Shirui Pan, et al. Time-llm: Time series forecasting by reprogramming large language models. In *International conference on learning representations*, volume 2024, pp. 23857–23880, 2024.

Dongbin Kim, Jinseong Park, Jaewook Lee, and Hoki Kim. Are self-attentions effective for time series forecasting? *Advances in neural information processing systems*, 37:114180–114209, 2024.

Dongbin Kim, Jaewook Lee, and Hoki Kim. Pattern-guided forecasting framework for metal price prediction with grouping decomposed series. *Financial Innovation*, 12(1):45, 2026.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

Yaxuan Kong, Yiyuan Yang, Yoontae Hwang, Wenjie Du, Stefan Zohren, Zhangyang Wang, Ming Jin, and Qingsong Wen. Time-mqa: Time series multi-task question answering with context enhancement. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 29736–29753, 2025.

Atli Kosson, Bettina Messmer, and Martin Jaggi. Rotational equilibrium: How weight decay balances learning across neural networks. In *Forty-first International Conference on Machine Learning*, 2024.

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 conference on empirical methods in natural language processing*, pp. 3045–3059, 2021.

Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *International conference on machine learning*, pp. 19730–19742. PmLR, 2023.

Jiafeng Lin, Yuxuan Wang, Jialong Wu, Huakun Luo, Zhongyi Pei, and Jianmin Wang. Thoth: Mid-training bridges llms to time series understanding. *arXiv preprint arXiv:2603.01042*, 2026.

Yun Luo, Zhen Yang, Fandong Meng, Yafu Li, Jie Zhou, and Yue Zhang. An empirical study of catastrophic forgetting in large language models during continual fine-tuning. *IEEE Transactions on Audio, Speech and Language Processing*, 2025.

Yuqi Nie, Nam H Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=Jbdc0vTOcol.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35: 27730–27744, 2022.

Qwen Team. Qwen3.8-max: A new bar for coding and cowork, August 2026. URL https://qwen.ai/blog?id=qwen3.8.

Chengsen Wang, Qi Qi, Jingyu Wang, Haifeng Sun, Zirui Zhuang, Jinming Wu, Lei Zhang, and Jianxin Liao. Chattime: A unified multimodal time series foundation model bridging numerical and textual data. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 12694–12702, 2025a.

Yilin Wang, Peixuan Lei, Jie Song, Yuzhe Hao, Tao Chen, Yuxuan Zhang, LEI JIA, Yuanxiang Li, and zhongyu wei. ITFormer: Bridging time series and natural language for multi-modal QA with large-scale multitask dataset. In *Forty-second International Conference on Machine Learning*, 2025b.

Andrew Robert Williams, Arjun Ashok, Étienne Marcotte, Valentina Zantedeschi, Jithendaraa Subramanian, Roland Riachi, James Requeima, Alexandre Lacoste, Irina Rish, Nicolas Chapados, and Alexandre Drouin. Context is key: A benchmark for forecasting with essential textual information. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=ih2WuBT1Fn.

Gerald Woo, Chenghao Liu, Doyen Sahoo, Akshat Kumar, and Steven Hoi. CoST: Contrastive learning of disentangled seasonal-trend representations for time series forecasting. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=PilZY3omXV2.

Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in neural information processing systems*, 34:22419–22430, 2021.

Haixu Wu, Tengge Hu, Yong Liu, Hang Zhou, Jianmin Wang, and Mingsheng Long. Timesnet: Temporal 2d-variation modeling for general time series analysis. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=ju_Uqw384Oq.

Stephan Xie, Ben Cohen, Mononito Goswami, Junhong Shen, Emaad Khwaja, Chenghao Liu, David Asker, Othmane Abou-Amal, and Ameet Talwalkar. Arfbench: Benchmarking multimodal time series reasoning for software incident response. In *1st ICLR Workshop on Time Series in the Age of Large Models*, 2026.

Zhe Xie, Zeyan Li, Xiao He, Longlong Xu, Xidao Wen, Tieying Zhang, Jianjun Chen, Rui Shi, and Dan Pei. Chatts: Aligning time series with llms via synthetic data for enhanced understanding and reasoning. In *Proceedings of the VLDB Endowment, 2025*, 2025.

Yao Yin, Zhenyu Xiao, Musheng Li, Yiwen Liu, Sutong Nan, Yiting He, Ruiqi Wang, Zhenwei Zhang, Qingmin Liao, and Yuantao Gu. Mmts-bench: A comprehensive benchmark for time series understanding and reasoning. *arXiv preprint arXiv:2602.08588*, 2026.

Zhihan Yue, Yujing Wang, Juanyong Duan, Tianmeng Yang, Congrui Huang, Yunhai Tong, and Bixiong Xu. Ts2vec: Towards universal representation of time series. In *Proceedings of the AAAI conference on artificial intelligence*, volume 36, pp. 8980–8987, 2022.

Haochuan Zhang, Chunhua Yang, Jie Han, Liyang Qin, and Xiaoli Wang. Tempogpt: Enhancing time series reasoning in multi-modal language model via quantizing embedding. *Neurocomputing*, 664:132035, 2026. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2025.132035.

Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 11106–11115, 2021.

Zhuohang Zhu, Haodong Chen, Qiang Qu, and Vera Chung. Fincast: A foundation model for financial time-series forecasting. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*, pp. 4539–4549, 2025.

A Benchmark Construction Details

All series in CiT-Bench are synthetic, so that the parameters of every event are known exactly. This appendix describes how the series, the events and the contexts are generated and how the generated scenes are checked. The anomaly detection set and the imputation and forecasting sets come from two generators, which share the seven base templates but differ in scale, noise and layout. Time steps are numbered from 0 as in the prompts, and an interval from step a through step b includes both endpoints. We write $U(a, b)$ for a uniform draw from the interval $[a, b]$ and $U\{a, \dots, b\}$ for a uniform draw from the integers a to b.

A scene is one generated series together with its events, and an instance pairs a scene with one context. In anomaly detection, a scene contains two candidate subsequences c_A and c_B and yields four instances, the two levels crossed with the two conditions. In condition A the context declares c_B normal, so the target is c_A, and condition B does the reverse. In imputation and forecasting, a scene contains one described event and yields two instances, one per level. All instances of a scene share the series, and instances that differ only in the level share the target. Each scene is drawn by rejection sampling until the quality checks of Appendix A.5 hold.

A.1 Base Waveforms and Generation Parameters

Three kinds of series occur. Periodic series are built from one of the seven one-cycle templates of Table 6, burst series serve the self-repetition type, and paired waves serve the delay type. Table 7 lists the generation parameters, and Table 8 defines the references that Level 2 contexts use, all of which can be read from the observations.

Periodic series. In anomaly detection, the usual signal u_t is the template scaled to unit variance over one cycle plus a linear drift, and the values are not standardized. In imputation and forecasting, it is generated in the units of the series as

$$u_t = c_0 + \beta \left(t - \tfrac{L-1}{2}\right) + \rho\, g(\phi_0 + t/P)\,, \tag{5}$$

where g is the template rescaled to $[-\frac{1}{2}, \frac{1}{2}]$ as a function of the phase in cycles, c_0 sets the level, the slope β sets the drift, ρ sets the amplitude, P is the period and L is the number of steps. In both generators the sampled phases repeat exactly from cycle to cycle, and in amplitude scenes anchored on the center line the initial phase is chosen so that at least one sample per cycle lies on the center line. A frequency change with the new period P' keeps this form for μ_t but replaces the phase $\phi_0 + t/P$ by one that advances by $1/P'$ per step during the event.

Burst series and paired waves. A burst series stays at a quiet level apart from irregular bursts, each of which leaves the quiet level and returns to it exactly, and the distances between bursts vary so that they do not recur at a fixed interval. In a paired-wave scene, series X is a smooth random wave plus noise, with no period and with shown values that need not span the whole range of the wave, and series Y is the same wave delayed by the usual delay d_0 plus independent noise, except where the context describes a change.

Table 6: One-cycle templates of the periodic series. Shape parameters are drawn per scene.

Template	One cycle
Sine	one sine cycle
Harmonic sum	a sine plus one or two harmonics, the k-th with relative amplitude $U(0.25, 0.6)/k$ and a random phase
Drifting sine	a sine, with the drift of Table 7 added to the series
Rectified sine	$\lvert \sin \pi\phi \rvert$
Exponential	a rise over the first 5% to 12% of the cycle, then a decay with a time constant of 15% to 30% of the cycle
Pulse train	one asymmetric Gaussian pulse
Bimodal	two Gaussian peaks, the second at 45% to 70% of the height of the first

Table 7: Generation parameters. The upper part applies to both generators, and the lower part only to the imputation and forecasting generator.

Parameter	Anomaly detection	Imputation and forecasting
Level and amplitude	periodic part with unit variance, S from 2.65 to 3.76	$c_0 \sim U(-50, 150)$, $\rho = 10^{U(1,2)}$
Usual period P	18 to 21 for clipping, otherwise 12 to 16, 18 for the harmonic sum and 12 or 18 for frequency changes	12 to 28, at least 18 for the harmonic sum, exceptions in Appendix A.2.2
Initial phase ϕ_0	$U(0, 1)$	$U(0, 1)$
Length	$L \sim U\{\max(160, 12P), \ldots, 256\}$	imputation L from 128 to 256 with a missing block of $W \in \{24, 32, 48\}$ steps, forecasting history $T \in \{128, 160\}$ and horizon $H \in \{24, 32, 48\}$
Drift	$\gamma\,(t - \frac{L-1}{2})/L$ with $\gamma = \pm U(0.6, 1.2)$ for the drifting sine and $\gamma \sim U(-0.3, 0.3)$ otherwise	$\beta = \pm U(0.5, 1)\,\rho/(L-1)$ for the drifting sine, $\beta = q_0$ for trend rate, 0 otherwise
Noise σ	$U(0.02, 0.15)$	$U(0.02, 0.04)\,S$
Shown precision	two decimals	fewest of one to three decimals with a rounding step of at most $\sigma/10$
Imputation and forecasting only		
Bursts	quiet level $U(-50, 150)$, height $U(0.45, 1)\,\rho$, raised-cosine taper over the first and last quarter, one peak shaped as a hump, a skewed hump, a rising ramp, a falling ramp or a plateau, or two peaks with probability 0.35 when at least 12 steps wide	
Two-peaked bursts	peaks at least 3 steps apart, dip between 40% of the height and 85% of the lower peak	
Burst spacing	8 to 28 quiet steps between consecutive bursts on the same side of a block, largest and smallest start-to-start distance at least 6 steps apart	
Paired waves	level $U(-50, 150)$, amplitude $10^{U(1,2)}$, white noise smoothed by a Gaussian kernel with standard deviation $U(2.5, 4)$ steps truncated at 12 steps and rescaled to $[0, 1]$ over a stretch longer than the series, usual delay $d_0 \in \{6, 8, \ldots, 16\}$, one σ for X and Y	

Table 8: References used by Level 2 contexts.

Reference	Series	Definition
Usual range S	periodic	highest minus lowest value of the periodic part at the phases that one cycle samples, which excludes the drift
	burst, wave	range of the noise-free observed series
Center line m_t	periodic	midpoint of these two values, following the drift
Envelopes	periodic	$m_t \pm S/2$
Usual interval	periodic	the period P
Usual delay d_0	wave	delay of Y behind X outside the event
Earlier burst	burst	the first or the second burst of the series

A.2 Events

Let u_t be the usual noise-free signal and μ_t the noise-free value with the event. The observed value is $x_t = \mu_t + \varepsilon_t$ with independent $\varepsilon_t \sim \mathcal{N}(0, \sigma^2)$, except that a bounding or clipping event records a reading beyond the bound as the bound. An event covers the steps s through e and has $D = e - s + 1$ steps. Outside this interval $\mu_t = u_t$, except that some frequency changes in anomaly detection leave the later pattern out of phase, as described below.

Table 9: Event types of the anomaly detection set. The two candidates of a scene share every parameter except the listed attribute, and the two clipping windows are drawn independently. The anchor o_t lies on the center line for the sine, the harmonic sum and the drifting sine and on the lower envelope otherwise, and m_s is the center line at the first held or clipped step.

Event	Change	Attribute pairs	Length D
Level shift	$\mu_t = u_t \pm a$	a/S: $\{0.5, 1\}$, $\{0.5, 1.5\}$, $\{1, 1.5\}$	$P/2$ to P, at least 6
Amplitude change	$\mu_t = o_t + f\,(u_t - o_t)$	f: $\{0.5, 1.5\}$, $\{0.5, 2.5\}$, $\{1.5, 2.5\}$	$2P$
Flat segment	$\mu_t = m_s \pm \ell$	$\ell/(S/2)$: $\{0.25, 0.5\}$, $\{0.5, 1\}$, $\{0.25, 1\}$	8 to 16
Clipping	readings beyond $m_s \pm b$ recorded as the bound	$b/(S/2)$: $\{0.25, 0.5\}$	window of P to $2.5P$
Frequency change	phase advances by $1/p$ per step	p: $\{4, 6\}$ at $P = 12$, $\{6, 9\}$ or $\{4, 9\}$ at $P = 18$	P
Duration	$\mu_t = u_t \pm a$ with $a \sim U(0.6, 1.2)\,S$	D/P: $\{0.5, 1\}$, $\{0.5, 2\}$, $\{1, 2\}$	6 to 36

A.2.1 Anomaly Detection

Table 9 defines the six event types. The two candidates of a scene share every parameter except the attribute that the context specifies, and their attribute values form one of the listed pairs. The only exception is clipping, where the two application windows, and hence the two answer lengths, are drawn independently. Except for clipping, the first candidate starts between $2P$ and $2.5P$ steps into the series, the second starts between $1.5P$ and $2P$ steps after the first ends, and at least $0.75P$ steps follow the second, all rounded down. For amplitude changes, these positions are the earliest possible starts, and each candidate begins at the first step within the next P steps where the usual signal meets the anchor. Clipping windows are separated by P to $2P$ steps, with at least $2P$ steps before the first window and after the second. The two candidates never overlap and are separated by at least 18 steps. Both frequency candidates repeat faster than the usual pattern. For $P = 18$ and a new period of $p = 4$ steps, a segment advances by 4.5 cycles in place of one, so the pattern after it continues half a cycle out of phase, and in all other frequency scenes the pattern returns in step. The target is the interval of the target candidate. For all types except clipping, it is the event interval of length D. For clipping, it runs from the first to the last sample that was actually clipped, including the unclipped samples between them. These intervals have 20 to 44 steps and are shorter than the application window in 16 of the 24 episodes, and the window itself never appears in a prompt.

A.2.2 Imputation and Forecasting

In imputation, the event covers the region of interest (RoI) inside the missing block, and at least two missing steps on each side of the RoI continue the usual pattern. In forecasting, the event starts at a step s with $T + 6 \le s \le T + H/2$ and lasts until the end of the window, so the RoI consists of the steps s through $T + H - 1$ and covers at least half of the window. The ground truth is the realized noisy value at every requested step, clipped for bounding, and it is stored without rounding. Table 10 defines the event types, and the following paragraphs give the rules that the table does not show.

Periods. Frequency and trend rate use $P \in \{12, 16, 20\}$, and amplitude uses a period of at most $\min(24, W - 4)$ steps in imputation and $\min(24, H - 4)$ steps in forecasting.

Amplitude. An amplitude change starts at a step where the noise-free signal lies on the anchor, so the scaled segment joins the usual signal without a jump. In imputation, the RoI spans exactly one period, so the step after it lies on the anchor again. In forecasting, the start is drawn from the anchor steps between $T + 6$ and $T + H/2$. Level 1 states the anchor line by its level at s and its slope per step, both rounded to two decimals.

Table 10: Event types of the imputation and forecasting sets. The change applies on the RoI, which in forecasting runs to the end of the window, and o_t and m_s are defined as in Table 9. The number of scenes per attribute value is given in the text.

Event	Change	Attribute values	RoI length in imputation
Shift	$\mu_t = u_t \pm fS$	$f \in \{\frac{1}{4}, \frac{1}{2}, \frac{3}{4}\}$	10 to 16
Amplitude	$\mu_t = o_t + k\,(u_t - o_t)$	$k \in \{0.5, 1.5\}$	P
Flat	$\mu_t = m_s \pm fS/2$	$f \in \{\frac{1}{4}, \frac{1}{2}, 1\}$	8 to 16
Bounding	readings beyond $m_s \pm fS/2$ recorded as the bound	$f \in \{\pm\frac{1}{4}, \pm\frac{1}{2}\}$	$\lfloor 0.6P \rfloor$ to P, within 8 to 20
Frequency	phase advances by $1/P'$ per step	$P'/P \in \{\frac{1}{4}, \frac{1}{3}, \frac{1}{2}, 2\}$	P if faster, $2P$ if slower
Trend rate	$\mu_t = u_t + (q - q_0)(t - s + 1)$	$q/q_0 \in \{\frac{1}{2}, 2, -1\}$	forecasting only
Self-repetition	replays the first or the second burst	first or second burst	width of the burst
Delay	Y follows X with delay d	$d/d_0 \in \{\frac{1}{2}, 2\}$	$W/4$ to $W - 4$, at least 8

Frequency. A faster change requires an integer P' of at least 4. In imputation, $D/P' - D/P$ is a whole number of cycles, so the pattern after the RoI is in step with the pattern before it. In forecasting, the change lasts until the end of the window, so no alignment is needed. In each set, $P'/P = \frac{1}{2}$, $\frac{1}{3}$, $\frac{1}{4}$ and 2 occur in 9, 2, 6 and 17 scenes.

Trend rate. Trend rate uses the six templates without built-in drift. The usual slope q_0 has magnitude $U(0.2, 0.4)\,S/P$, rounded to 0.02, so the center line moves by 0.2 to 0.4 usual ranges per cycle, and its sign alternates across scenes. The new slope q is rounded to two decimals.

Bounding. A negative f places the bound past the center line. The fractions follow a fixed schedule, and a fraction that repeatedly fails the quality checks is replaced by the next one in the schedule. The imputation set uses $f = \frac{1}{4}$, $\frac{1}{2}$, $-\frac{1}{4}$ and $-\frac{1}{2}$ in 5, 4, 9 and 6 scenes, and the forecasting set uses $f = -\frac{1}{4}$, $\frac{1}{4}$ and $\frac{1}{2}$ in 9, 6 and 5 scenes.

Self-repetition. In imputation, two to four bursts precede the missing block and up to two follow it, and the rest of the block stays at the quiet level. The first and the second burst have the same width, so the length of the RoI does not reveal which one is meant. In forecasting, four or five bursts lie in the history. From step s the series replays its course from the first step of the referenced burst, and the replayed stretch can include the quiet steps after the burst and the start of the next one. In both tasks the referenced burst is two-peaked in 16 of the 33 scenes.

Delay. A change of delay is admitted only where the joints are smooth, with the second difference of the noise-free Y around each joint within the 90th percentile of the second differences of the wave, and the new delay moves Y by at least $0.15\,S$ on average over the RoI. In imputation, X is shown in full, and enough of Y is observed before and after the missing block for the usual delay to be estimated. In forecasting, X is shown from step H through step $T + H - 1$ and Y from step 0 through step $T - 1$, so X covers the whole window for either direction of the change.

Rounding. Levels, bounds and shift amounts are chosen so that every number stated in a Level 1 context equals the value used to generate the series, with no rounding error.

A.3 Allocation by Type

Table 11 lists the number of scenes and instances of every event type. Each task has 400 instances, and each combination of level and category has 100. In anomaly detection, a combination therefore holds 50 A/B pairs.

Table 11: Scenes and instances per event type, given as scenes / instances. An anomaly detection scene yields four instances, and an imputation or forecasting scene yields two.

Category	Event type	Anomaly detection	Imputation	Forecasting
Magnitude	Shift	12 / 48	24 / 48	20 / 40
	Amplitude	14 / 56	28 / 56	20 / 40
	Flat	12 / 48	24 / 48	20 / 40
	Bounding or clipping	12 / 48	24 / 48	20 / 40
	Trend rate			20 / 40
Temporal	Frequency	25 / 100	34 / 68	34 / 68
	Duration	25 / 100		
	Self-repetition		33 / 66	33 / 66
	Delay		33 / 66	33 / 66
Total		100 / 400	200 / 400	200 / 400

Within each event type, the remaining factors follow fixed schedules over the scene index. The template, the direction of the change, the order of the two candidates in anomaly detection, and the block length, the history length and the horizon in imputation and forecasting each cycle through their values, so the counts of any two values of a factor differ by at most two scenes.

A.4 Level 1 and Level 2 Contexts

The two levels of a scene describe the same attribute of the same event. Level 1 states the attribute with explicit values that need no estimate from the observations, such as a value in the units of the series, a scaling factor with its anchor line, a number of steps or a step index. Level 2 states it relative to one of the references of Table 8, which must first be estimated from the observations. A context that refers to the usual range, the envelopes or the usual interval defines that reference in its closing sentences, the delay contexts define the usual delay, and the prompt instructions describe the bursts.

Anomaly detection. Every context begins with three fixed sentences about the series, describes the deviation that counts as normal, and asks for any other deviation (Figure 6). The description gives the attribute as a closed interval, the nominal value plus or minus 0.1 of the reference for magnitude attributes and plus or minus 15% for temporal attributes, in units or steps at Level 1 and as a percentage of the reference at Level 2. Table 12 shows one description per event type and level with its reference, and clipping contexts also state that the answer runs from the first to the last clipped sample.

Imputation and forecasting. Imputation contexts describe the change in the past tense over "From step s through step e", except that the Level 1 amplitude context gives it as an instruction to scale the deviations. Forecasting contexts describe it in the future tense from "From step s onward". Table 13 shows how the two levels state each event type.

A.5 Quality Control

Scenes are generated by rejection sampling, so all parameters of a scene are redrawn until the conditions of Table 14 or Table 15 hold. Every released scene satisfies all conditions, and the last column of Table 15 gives the most extreme value in the released sets, rounded to the digits shown. In anomaly detection, the matrix profile (cite: Yeh et al. "Matrix Profile I: All Pairs Similarity Joins for Time Series: A Unifying View That Includes Motifs, Discords and Shapelets." ICDM 2016.) gives the z-normalized Euclidean distance of each subsequence of length P to its nearest neighbor that starts at least $\lfloor P/2 \rfloor$ steps away.

In imputation and forecasting, the context-ignoring values are the usual signal, the quiet level for self-repetition, or X at the usual delay for delay, and both sets of trajectories are scored with the RCRPS of Appendix C. An independent validation repeats the context and burst checks on the released sets with fresh draws and its own thresholds, and it also checks the layout, the step numbers in every context, the prompt text and the delay joints. Across the remaining event types, the smallest

Table 12: Example descriptions of the normal candidate in anomaly detection, the text after "must NOT be reported:", with the reference of each attribute. The descriptions are quoted from the released prompts. The amplitude example comes from a scene anchored on the center line. Scenes anchored on the lower envelope name the lower envelope in its place and add a sentence that defines it.

Event	Reference	Level 1	Level 2
Level shift	usual range S	a segment where the whole pattern shifted upward; the shift was between 2.90 and 3.55 units.	. . . the shift was between 90% and 110% of the usual range.
Amplitude change	usual range S	a segment whose peak-to-trough range was between 3.96 and 4.53 units.	. . . between 140% and 160% of the usual range. Its centre line, shape and timing were unchanged; its usual amplitude resumed afterwards.
Flat segment	$S/2$	a segment where the underlying signal was held at a constant level between 0.67 and 1.02 units.	. . . held at a constant level instead of following its usual pattern; at the first step, that level was between 40% and 60% of the way from the centre line to the upper envelope.
Clipping	$S/2$	a clipping episode with a constant cap between -0.75 and -0.42 units.	. . . a constant cap between 15% and 35% of the way from the centre line to the lower envelope, measured at the episode's first clipped sample.
Frequency change	usual interval P	a segment where the pattern's repetition interval was between 5.1 and 6.9 steps.	. . . between 42.5% and 57.5% of the usual repetition interval.
Duration	usual interval P	an upward level shift lasting between 13.6 and 18.4 steps.	. . . lasting between 85% and 115% of the usual repetition interval.

Table 13: How the two levels state each imputation and forecasting event. Quotes follow the imputation contexts, except for trend rate, which exists only in forecasting. Forecasting states the same content in the future tense, with small differences in wording.

Type	Level 1	Level 2
Shift	"shifted upward by $\langle a \rangle$ units"	a quarter, half or three quarters of the usual range
Amplitude	the factor k and the level of the anchor line, with its slope when the anchor drifts	the envelope distance becomes half or one and a half times the usual range
Flat	"held at the constant value $\langle v \rangle$"	a quarter of the way or midway from the center line to an envelope, or at the envelope, "as of step s"
Bounding	"The cap was at $\langle b \rangle$ units"	a position relative to the center line and an envelope, "as of step s"
Trend rate	"will increase by $\langle q \rangle$ units per step"	twice or half the observed trend, or the observed trend reversed
Frequency	"repeated every $\langle P' \rangle$ steps"	half, a third, a quarter or twice the usual interval
Self-repetition	"repeated the pattern it showed from step $\langle a \rangle$ through step $\langle a + w - 1 \rangle$"	"repeated the pattern of its first burst" or of its second burst
Delay	"the delay was $\langle d \rangle$ steps" and the steps of X that Y repeats	"the delay was half its usual value" or twice

ratio of the context check lies between 2.88 and 5.53 in imputation and between 2.48 and 5.70 in forecasting.

The generator also keeps every joint continuous unless the change itself is a jump, as for shift and flat. An amplitude change starts on its anchor, a frequency change keeps the phase, bursts begin and end at the quiet level, and delay joints satisfy the smoothness rule of Appendix A.2.2. Outside the missing block and before the window, the observations equal the usual signal plus noise, so the change can be inferred only from the context.

Table 14: Quality conditions of the anomaly detection set.

Check	Event types	Condition
Candidate discord	level shift, flat segment, frequency, duration	The largest matrix-profile value over subsequences that overlap a candidate is at least 1.3 times the largest value over subsequences that overlap neither candidate.
Amplitude size	amplitude change	$\lvert f-1\rvert\, S$ is at least 5σ.
Clipped samples	clipping	At least four samples are clipped, and the answer interval contains at least three unclipped samples.
Context intervals	all	Each interval contains the attribute of exactly one candidate at both levels, the two intervals of a scene are disjoint, and the Level 1 and Level 2 intervals agree within 0.005 units or steps.

Table 15: Quality conditions of the imputation and forecasting sets. An empty validation cell marks a check that the validation does not repeat. The last column gives the smallest value in the released imputation and forecasting sets, taken from the validation where it repeats the check.

Check	Event types	Generation	Validation	Released, imputation / forecasting
Visible change	all except bounding	The mean of $\lvert\mu_t - u_t\rvert$ over the RoI is at least 2σ.		2.18σ / 2.35σ
Visible clipping	bounding	The clipped and the unclipped noisy readings differ by at least 2σ on average over the RoI, and the RoI has at least four clipped and three unchanged readings.		2.02σ with 4 and 3 / 2.05σ with 4 and 3
Visible trend	trend rate	The usual trend over one cycle is at least 4σ.		– / 7.09σ
Context necessity	all	The RCRPS of 25 trajectories around the context-ignoring values is at least 2.2 times, and at least 0.006 above, that of 25 trajectories around the true noise-free values.	The same with fresh draws, at least 2 times and 0.005 above.	ratio 2.50 for amplitude, difference 0.023 / ratio 2.39 for trend rate, difference 0.006
Correct burst	self-repetition	The context check also holds with the noise-free course of any other observed burst in place of the referenced one. In forecasting, this covers every other burst whose replay fits in the history.	A copy of any other burst taken from the observations scores at least 1.5 times worse than a copy of the correct one.	ratio 2.55 / 1.81

B Experimental Setup

B.1 Models

We evaluate ten models in three groups, open-weight LLMs, time series LLMs and closed models (Table 3). The seven models with public weights run in bfloat16 on a single NVIDIA H200 GPU. Models served with vLLM use a context limit of 16,384 tokens, which holds every prompt together with the output budget, except ChatTime-7B, which keeps its native limit of 4,096 tokens. GPT-4o mini and Gemini 3.1 Flash-Lite were queried on 25 September 2026, and Claude Haiku 4.5 on 26 September 2026.

B.2 Time Series Input

The open-weight LLMs, Thoth-30B and the closed models receive the released prompts of Appendix B.3 unchanged, so the series is given to them as text. Thoth-30B is a time series LLM built on Qwen3-30B-A3B-Instruct-2507, but it has no separate series input. ChatTS-14B, ITFormer-7B and ChatTime-7B have a native series input, which we use. For these three models only the series

part of the prompt changes. The sentence that introduces the series states the number of points and the index range, for example "Here is the time series (245 points, indices 0-244):", and the values enter through the native input of the model. The context, the instructions and the output format stay as released, so the answers are parsed and scored exactly as for the other models.

ChatTS-14B. The series bodies are replaced by the `<ts><ts/>` placeholder of ChatTS, and the values are passed to its time series encoder through the official processor, which normalizes each series and adds its offset and scale as text next to the placeholder. The two series of a delay scene follow the multivariate input format of ChatTS, with one placeholder per series, so each series is normalized separately. Missing steps are filled with the last observed value in the encoder input. This model was run on a separate server with the released package and Hugging Face Transformers.

ITFormer-7B. The series enters through the ITFormer encoder, whose fused output fills the 25 placeholder positions of the chat template of the frozen Qwen2.5-7B-Instruct, with values in their original units and right padding to a multiple of 60 steps that the text declares as not part of the series. Missing steps are filled by linear interpolation for the encoder only, the two series of a delay scene enter as two channels on a common step axis, and the text states which steps are missing or filled.

ChatTime-7B. The official ChatTime tokenizer scales all values of an instance by their minimum and maximum, with one scaling shared by the two series of a delay scene, and maps each value to a value token, with a separate token for missing steps. The prompt follows the official analysis template of ChatTime, with the system and user text as the instruction and the value tokens as the input, and the text states the scaling so that a value token can be converted back to the units of the series.

B.3 Prompts

Every instance is presented as a chat with one system message and one user message, and the text models receive exactly these messages. The system message names the task, and the user message gives the task, the context, the series and the output format. The series is written as index:value pairs, sixteen per line. Values have two decimals in anomaly detection and one or two decimals in imputation and forecasting, where missing steps appear as NaN. Figures 6 and 7 show a complete anomaly detection prompt and a complete imputation prompt with the series shortened. A forecasting prompt has the same layout as an imputation prompt. It introduces the series as past observations, asks to forecast the steps of the window using only the past observations and the context, states that the usual signal includes any gradual drift, and replaces the sentence about the steps outside the interval by "Before step s, continue the usual signal and noise without restarting the pattern." Self-repetition prompts replace the three sentences about the usual signal with sentences about the quiet level and the bursts, and delay prompts list both series and ask for the values of series Y (Figure 8).

B.4 Generation Settings

All models sample with temperature 1.0 and generate at most 2,048 new tokens per response. Except for Claude Haiku 4.5, they use top-p 1.0 without top-k truncation. The prompts ask for the answer only, and a response that contains reasoning is cut at the same limit. Models served with vLLM use no repetition penalty, and every request sets its sampling parameters explicitly, so the default sampling settings distributed with the models, such as top-k 20 for Qwen3.8-27B, are not applied. ChatTS-14B was sampled with Hugging Face Transformers and a repetition penalty of 1.05, the only model with such a penalty. ITFormer-7B was sampled with Hugging Face Transformers for anomaly detection and with vLLM on precomputed input embeddings for imputation and forecasting, with the same sampling parameters in both cases. Claude Haiku 4.5 receives only the temperature, because its API rejects temperature and top-p in the same request, so top-p and top-k stay at their API defaults.

Thinking is disabled for every model that has a thinking mode. The chat templates of the two Qwen models are rendered with thinking disabled, which inserts an empty thinking block, and the chat template of Gemma4-31B disables thinking by default. Gemini 3.1 Flash-Lite runs with a thinking budget of 0 and reported no thought tokens. Claude Haiku 4.5 runs without extended thinking,

System

```
You are a useful time series anomaly detection assistant.
```

User

```
I have a time series anomaly detection task for you.

<context>
The following time series follows a recurring periodic pattern with
ordinary background noise.
The series contains segments that deviate from the usual signal.
Time intervals and numerical ranges include both endpoints.
Deviations matching the following description are part of normal
operation and must NOT be reported:  a segment where the whole pattern
shifted upward; the shift was between 2.90 and 3.55 units.  Its shape
and timing were unchanged.
Report any other deviation from the usual signal as an anomaly.
The shift describes the underlying pattern, excluding background noise.
</context>

Here is the time series in index:value format:
<series>
0:-0.02, 1:0.58, 2:1.03, 3:1.42, 4:1.66, 5:1.61, 6:1.47, 7:1.03, ...,
15:1.07
...
240:1.37, 241:1.03, 242:0.52, 243:-0.20, 244:-0.85
</series>

Report the one deviation that the context does not describe as normal.
Use zero-based indices of the series (both endpoints included).  Report
the full interval affected by the deviation.

Respond ONLY with the JSON object {"start":  <int>, "end":  <int>}.  Do
not explain.  Output plain text only:  no Markdown and no code fences.
```

Figure 6: Prompt of a Level 1 anomaly detection instance with level shifts. The series is shortened.

which is off unless a request enables it. Thoth-30B, ChatTS-14B, ITFormer-7B, ChatTime-7B and GPT-4o mini have no thinking mode.

Each draw round of an instance has a fixed seed, obtained by hashing the task, the instance and the index of the round, and the released evaluation code implements this rule. In vLLM and for GPT-4o mini, all responses of a round come from one request with this seed. Gemini returns one response per request, so the j-th draw of a round uses the seed plus j. The Hugging Face Transformers runs of ITFormer-7B seed the generator with the same value before each draw. The Claude API also returns one response per request and has no seed parameter, so the draws of Claude Haiku 4.5 are not seeded. Seeds of the OpenAI and Gemini APIs are best effort, and seeded sampling in vLLM depends on the composition of the batch, so a rerun follows the same protocol but does not reproduce every response. Additional draws only replace invalid responses and never depend on correctness, and no majority voting or answer selection is used (Appendix C). In the anomaly detection runs of Gemma4-31B, Qwen3.8-27B, Qwen3.5-9B and Thoth-30B, all five draws of every instance were generated in advance with these seeds, and the first valid draw is scored, so the selection rule is the same.

C Evaluation Metrics

Answer parsing. We parse every draw with fixed rules, without an LLM judge or manual correction. For anomaly detection, we read the first occurrence of `"start": <int>, "end": <int>` in the output, and the draw is valid if $0 \leq start \leq end < L$. An output that contains more than one such pair is not rejected, and only the first pair is scored. For imputation and forecasting, the series is shown as `index:value` pairs and the answer is requested as one `(index, value)` pair per line. The parser accepts `(index, value)`, `index, value`, and `index:value` anywhere in the output, keeps the last value when an index repeats, and ignores other text. A draw is

System

```
You are a useful time series imputation assistant.
```

User

```
I have a time series imputation task for you.

Here is some context about what happens during the missing interval.
<context>
From step 61 through step 76, the whole pattern was shifted upward by
3.07 units; its shape and timing were unchanged.
</context>

Here is the time series in index:value format, with the missing steps
marked NaN:
<series>
0:-8.09, 1:-8.72, 2:-8.35, 3:-7.55, 4:-5.99, 5:-3.55, 6:-1.13, ...,
15:-5.83
...
48:-7.77, ..., 54:-1.35, 55:NaN, 56:NaN, ..., 78:NaN, 79:-6.79, ...
...
208:-8.34, 209:-8.75, 210:-8.99, 211:-7.63, 212:-5.65, 213:-3.51,
214:-1.38
</series>

Steps 55-78 (24 values) are missing.
Infer the usual signal and the noise strength from the observations.
The usual signal is the repeating pattern without the change described
in the context.
Outside the interval described in the context, continue the usual signal
and noise without restarting the pattern.
Noise is independent, zero-mean Gaussian with unchanged strength.
Intervals include both endpoints.
Return one plausible realization including noise, rather than a
noise-free average.

Respond ONLY with the 24 values, one per line in (index, value) format
with 2 decimal places.  Do not explain.
```

Figure 7: Prompt of a Level 1 imputation instance with a shift. The series is shortened.

valid if every required index appears with a finite value. The requested number of decimals is not enforced. A value followed on the same line by +, $\star$, or / and another number is treated as an arithmetic expression and makes the draw invalid. A minus sign is not treated as an operator, since it cannot be distinguished from a negative value.

IoU. Classical anomaly detection metrics score a per-step anomaly sequence, whereas an LLM returns text and the target in Section 3.1 is a single segment. We therefore ask the model to return the start and end indices of the segment as a JSON object, {"start": <int>, "end": <int>}, with zero-based indices and both endpoints included, and score its overlap with the target segment. An output is invalid if it does not contain such a pair of indices, if an index lies outside the series, or if the start index is later than the end index. We redraw invalid outputs, up to five draws in total, and score the first valid one. An instance without a valid output is counted as incorrect. For a valid segment $\hat{A}$ and the target A, we compute the intersection over union

$$\mathrm{IoU}(\hat{A}, A) = \frac{|\hat{A} \cap A|}{|\hat{A} \cup A|}, \tag{6}$$

where $|\cdot|$ counts time steps. An instance is correct if $\mathrm{IoU}(\hat{A}, A) \geq 0.5$. We report accuracy, the fraction of correct instances, and pair consistency, the fraction of A/B pairs in which both instances are correct.

RCRPS. For imputation and forecasting, each draw is one plausible realization of the required steps, and a draw is valid if it gives a value for every required step. We draw 25 outputs and, while

Self-repetition, imputation

```
Infer the quiet level, the bursts and the noise strength from the
observations.
The series stays at a quiet level apart from occasional short bursts;
each burst leaves the quiet level and returns to it, with one or two
peaks.
Outside the interval described in the context, the missing steps stay at
the quiet level.
```

Delay, forecasting

```
Here are the observations of two series in index:value format.  Series X
is observed from step 24 through step 151; series Y is observed through
step 127:
<series name="X">
24:73.74, 25:71.88, ..., 151:77.39
</series>
<series name="Y">
0:79.75, 1:77.50, ..., 127:84.92
</series>

Forecast series Y for steps 128-151 (24 values) using only the
observations and context.
Infer the usual delay and the noise strength from the observations.
Before step 135, the delay has its usual value.
...
Respond ONLY with the 24 values of series Y, one per line in (index,
value) format with 2 decimal places.  Do not explain.
```

Figure 8: Lines that differ in self-repetition and delay prompts. In forecasting self-repetition prompts, the first of these lines ends with "from the past observations", and the third reads "Before step s, the series stays at its quiet level.". Imputation delay prompts state that series X is complete and series Y has missing steps marked NaN.

fewer than 25 are valid, up to three further rounds of five, and form an ensemble from the first $n = 25$ valid draws. For step t with ensemble values $x_{1,t}, \ldots, x_{n,t}$ and ground truth y_t, we use the fair CRPS

$$\mathrm{CRPS}_t = \frac{1}{n}\sum_{i=1}^{n}|x_{i,t} - y_t| - \frac{1}{2n(n-1)}\sum_{i\neq j}|x_{i,t} - x_{j,t}|. \tag{7}$$

Following CiK (**?**), we weight the RoI $\mathcal{R}$ and the remaining evaluated steps $\mathcal{M} \setminus \mathcal{R}$ equally and normalize by the range Δ of the observed values of the target series,

$$\mathrm{RCRPS} = \frac{1}{2\Delta}\left(\frac{1}{|\mathcal{R}|}\sum_{t\in\mathcal{R}}\mathrm{CRPS}_t + \frac{1}{|\mathcal{M}\setminus\mathcal{R}|}\sum_{t\in\mathcal{M}\setminus\mathcal{R}}\mathrm{CRPS}_t\right). \tag{8}$$

For imputation, $\mathcal{M}$ is the missing block, and the second term pools the steps before and after the RoI. For forecasting, $\mathcal{M}$ is the forecast horizon, and since the change lasts until the end of the horizon, the second term covers the steps before the RoI. An instance without 25 valid draws after 40 draws receives a score of 1. We report the mean RCRPS over all instances, including these failures, together with coverage, the fraction of instances with a complete ensemble.

D Tokenization Details

Embeddings. For a series $\mathbf{x} = (x_1, \ldots, x_N)$, the scale exponent k is the smallest integer with $10^k \geq \max_\tau |x_\tau|$, and the normalized values are $z_\tau = x_\tau/10^k \in [-1, 1]$. A value z is embedded on a grid of $B + 1 = 201$ points $g_i = (2i - B)/B$ by interpolating a trainable table $\mathbf{E} \in \mathbb{R}^{(B+2)\times d}$, and the exponent is embedded as a continuous function of k with trainable $\mathbf{b}, \mathbf{d} \in \mathbb{R}^d$,

$$\mathbf{e}(z) = (1 - f)\,\mathbf{E}_b + f\,\mathbf{E}_{b+1}, \qquad \mathbf{e}_s(k) = \mathbf{b} + \tanh(k/\kappa)\,\mathbf{d}, \tag{9}$$

where $g_b \le z \le g_{b+1}$, $z = (1-f)\,g_b + f\,g_{b+1}$, and κ is a fixed temperature. This two-hot embedding of z incurs no quantization error while the table keeps 201 rows, the magnitude of the series is retained in k, and an exponent absent from training is interpolated between its neighbors rather than mapped to an untrained row. A missing observation is embedded as the additional row $\mathbf{E}_{B+1}$.

Scale selection. The exponent k is evaluated by direct comparison, starting from $\lfloor \log_{10} m \rfloor$ and stepping until $10^{k-1} < m \le 10^k$, rather than by rounding a logarithm, which can yield $10^k < m$ under floating-point error. The floor $k \ge -2$ can only increase k, so $|z| \le 1$ holds for every observed value. The set of values entering m depends on the task. For forecasting and imputation it is the observed input only, since the answer is text and need not lie on the grid. For multi-series tasks it is the union of all presented series. For the arithmetic tasks the answer is included in 80% of examples and deliberately excluded in the remaining 20%, so that the answer leaves the input range (Appendix G).

Grid. The offset f is stored in single precision. Each value position additionally carries a nearest-grid-point index $b + \mathbf{1}[f \ge 0.5]$ that identifies the position as a value word, but the embedding reads only (b, f). Values with $|z| > 1$ cannot arise for observed points. Were they to occur, the end bins would absorb them ($z > 1 \mapsto (b, f) = (B-1, 1)$, $z < -1 \mapsto (0, 0)$). The round-trip error of the nearest-grid-point index alone is at most half a grid step, $0.005 \cdot 10^k$ in original units, whereas the two-hot input has none.

Block layout. A series of N points is rendered into the prompt as a block of $N + 3$ tokens,

$$\mathbf{H}(\mathbf{x}) = \left[\, \mathbf{e}_{\text{start}},\ \mathbf{e}_s(k),\ \mathbf{e}(z_1),\ \ldots,\ \mathbf{e}(z_N),\ \mathbf{e}_{\text{end}} \,\right] \in \mathbb{R}^{(N+3)\times d}. \tag{10}$$

The delimiters $\mathbf{e}_{\text{start}}$ and $\mathbf{e}_{\text{end}}$ are the frozen embeddings of the image-boundary tokens of the backbone, which was pre-trained as a vision-language model in which these two tokens delimit non-text content, so reusing them introduces no parameters. At the embedding layer, text positions are looked up in the frozen table, value positions are replaced by $\mathbf{e}(z)$ and the scale position by $\mathbf{e}_s(k)$ of Equation 9. The language-model head is unchanged. The normalization subtracts no mean and rescales only by a power of ten, so sums, differences, and constant shifts of raw values correspond to the same operations on z. The linear dependence of $\mathbf{e}_s$ on k follows the positional biases of **?**, and the saturation follows the number encoding of Golkar et al. (2023).

Scale word. The temperature κ of Equation 9 is fixed at 3 and is not trained. Over the exponents that occur in training, $k \in \{-2, \ldots, 6\}$, $\tanh(k/\kappa)$ ranges from -0.58 to 0.96, so no two exponents share an embedding and the embeddings of the two largest exponents differ least. The initialization $\mathbf{b}$ = embedding of the digit token "1" and $\mathbf{d} = \mathbf{0}$ makes $\mathbf{e}_s(k)$ equal to a valid text embedding for every k at the first step. The gradient with respect to $\mathbf{d}$ is scaled by $\tanh(k/\kappa)$ and therefore vanishes for $k = 0$, so the direction $\mathbf{d}$ is learned from the examples with $k \ne 0$ only.

Vocabulary. Table 17 lists the token ids and Table 16 the trainable tensors. The scale word owns no table row, since its embedding is computed from $(\mathbf{b}, \mathbf{d})$. The table therefore has 202 rows while 204 vectors are trained. In the pre-trained backbone the norms of the two boundary embeddings are 0.996 and 0.989, comparable to those of the chat-template delimiter tokens (0.85 and 0.84) and to the median text row (0.929), whereas the audio-boundary tokens, which were not used in pre-training, have norm 0.23.

Table 16: Trainable parameters of NEOTS for Qwen3.8-27B ($d = 5{,}120$).

Tensor	Shape	Parameters
Value and missing-value rows $\mathbf{E}$	$202 \times 5{,}120$	1,034,240
Scale base $\mathbf{b}$	5,120	5,120
Scale direction $\mathbf{d}$	5,120	5,120
Total	204 vectors	1,044,480

Table 17: Token ids of the time-series tokenizer. The time-series ids follow the text vocabulary of Qwen3.8-27B.

Token	Id	Embedding
image-boundary start, end	248,053, 248,054	frozen backbone rows
value words 0–200	248,320–248,520	rows $\mathbf{E}_0, \ldots, \mathbf{E}_{200}$, read through Equation 9
missing value	248,521	row $\mathbf{E}_{201}$
reserved	248,522	never occurs
scale word	248,523	$\mathbf{e}_s(k)$, Equation 9

Number formatting. Text numbers use $\max(0, 4-k)$ decimals, which gives four significant digits relative to the largest absolute value of the series, omit trailing zeros, never print -0, and write non-finite values as a single missing-value token. A decimal list accompanying a block lists exactly one entry per value word, with missing values written as that token, so that text and block remain aligned. An optional index line directly above a block carries a date axis or an integer axis. A date axis is either listed, as in "Index: 2021-01-28, 2021-02-04, …", or compressed, as in "Index: from 2025-05-10, daily". Sub-daily stamps include the time of day. An integer axis is listed, as in "Index: 1, 2, …".

Prompt format. Examples use the chat template of the backbone with a system line drawn from a pool of 18, an empty reasoning block, and the answer. The label span begins at the first answer token and ends with the end-of-turn token. The empty reasoning block belongs to the prompt and receives no loss. Training and evaluation share a single block-assembly routine.

E Training and Inference Details

Backbone. The backbone is the decoder-only language model of Qwen3.8-27B with hidden size $d = 5{,}120$ and 26.90B parameters including the output head, all frozen. Its vision encoder is discarded. The 1,044,480 trainable parameters of Table 16 amount to 0.004% of the backbone.

Packing. Multiple examples are not packed into one sequence. The hybrid recurrent and attention layers of the backbone admit only a padding mask and cannot reset their state at example boundaries, so packing would let examples leak into one another. Examples are instead right-padded within length-sorted buckets, at a padding overhead of approximately 2–3%.

Table 18: Training configuration.

Backbone	Qwen3.8-27B language model, frozen (26.90B parameters including output head), vision encoder discarded
Trainable parameters	1,044,480 (Table 16), kept in single precision
Examples	3,600,000 (Table 21), one epoch
Batch	128 examples (32 per device $\times$ 4 devices)
Steps / tokens	28,125 / 1.63×10^9 (prompts included)
Optimizer	AdamW, $\beta = (0.9, 0.95)$, $\epsilon = 10^{-8}$, weight decay 0
Learning rate	10^{-3}, constant, no warm-up
Gradient clipping	global norm 1.0
Norm penalty	$\lambda = 0.1$, $\tau = 1.0$, applied once per optimizer step, before clipping
Scale word	$\kappa = 3$, fixed
Curriculum	first 100,000 examples are transcription (Appendix H)
Precision	bf16 parameters, fp32 gradient reduction
Sequence handling	no packing, length-sorted buckets with right padding, maximum length 1,752
Activation checkpointing	on
Hardware / time	4 $\times$ NVIDIA H200 (141 GB), FSDP, 39.9 h
Seed	1234

Norm penalty versus weight decay. The target norm $\tau = 1$ of Equation 4 is chosen to match the median norm of the text embeddings of the backbone, 0.929. Unlike weight decay, the penalty is inactive for a vector already within the target norm, and it acts on the new vectors only, since the backbone is frozen and no weight decay is applied. A preliminary run without any penalty, but otherwise identical, reached a mean ℓ_2 norm of 9.9 and a maximum of 13.1 over the trainable vectors after 20,000 steps, an order of magnitude above the text embeddings. Weight decay is not a substitute at this learning rate. Under the AdamW equilibrium analysis of Kosson et al. (2024), a parameter vector trained with learning rate η and decay coefficient λ_{wd} settles at $\|\mathbf{v}\|^2 \approx \eta C/(2\lambda_{\mathrm{wd}})$, where $C \approx d$ under the assumption of unit per-coordinate Adam updates. With $\eta = 10^{-3}$ and $d = 5{,}120$, holding $\|\mathbf{v}\| \approx 0.93$ would require $\lambda_{\mathrm{wd}} \approx 3$, whereas the conventional $\lambda_{\mathrm{wd}} = 0.1$ equilibrates near 5.

Training dynamics. The training loss is stable throughout the run. Averaged over windows of 2,000 steps, it lies in $[0.50, 0.57]$ after the transcription segment (0.60 within it). The penalty remains active: the mean ℓ_2 norm of the 204 trainable vectors stays in $[0.871, 0.930]$, the maximum peaks at 1.006, at most nine vectors exceed τ at any logged step, and the final norms are $\|\mathbf{b}\| = 0.998$ and $\|\mathbf{d}\| = 0.997$. The gradient norm never exceeded 5.7.

Inference. At test time a series is rendered exactly as in training. It enters as a block (Equation 10) and, unless stated otherwise, is also written out as a decimal list, with the order of the two decided by the same pseudo-random rule as in training. The scale k is recomputed from the given history. Decoding settings are reported with each experiment.

F Results on Context is Key

Qwen3.8-27B+NeoTS is evaluated on Context is Key under the protocol of the comparison in Section 1, with the series entering as a block and as decimal text as in training. Figure 9 places the result among the general-purpose and time-series language models of that comparison, and Table 19 isolates the two comparisons of interest. Against the frozen backbone with decimal text alone, the time-series words lower the mean RCRPS from 0.136 to 0.127. Against Gemma4-31B, the best general-purpose model of the comparison at 0.132, Qwen3.8-27B+NeoTS is the best result among all compared models.

Table 19: Context is Key at 27B (RCRPS, lower is better). Gemma4-31B has no time-series words.

		Input		
Model	TS words	Text	Block	RCRPS
Gemma4-31B	–	✓	×	0.132
Qwen3.8-27B	untrained	✓	×	0.136
Qwen3.8-27B	trained	✓	✓	0.127

Fine-tuning at 4B. Table 20 completes the comparison of Figure 3 with Qwen3.5-4B. Two fully fine-tuned backbones are included, one trained on CiT-Corpus as decimal text and one trained with the time-series words and the block presentation. In both cases a share of the samples on Context is Key cannot be parsed as forecasts, so no score can be computed, whereas the frozen backbone with the time-series words improves on the untrained backbone.

G Training Corpus

G.1 Overview

The corpus of Section 4.1 is designed to connect series with language rather than for any downstream application. Its 31 tasks are divided by source: the six tasks of the continuing family use windows of the Chronos corpus (Appendix G.3, 900,000 examples) and the 25 tasks of the other families use generated waveforms (Appendix G.4, 2,500,000 examples). Instructions come from templated pools (Appendix G.5). Throughout this appendix an example presents its input as *block and text*

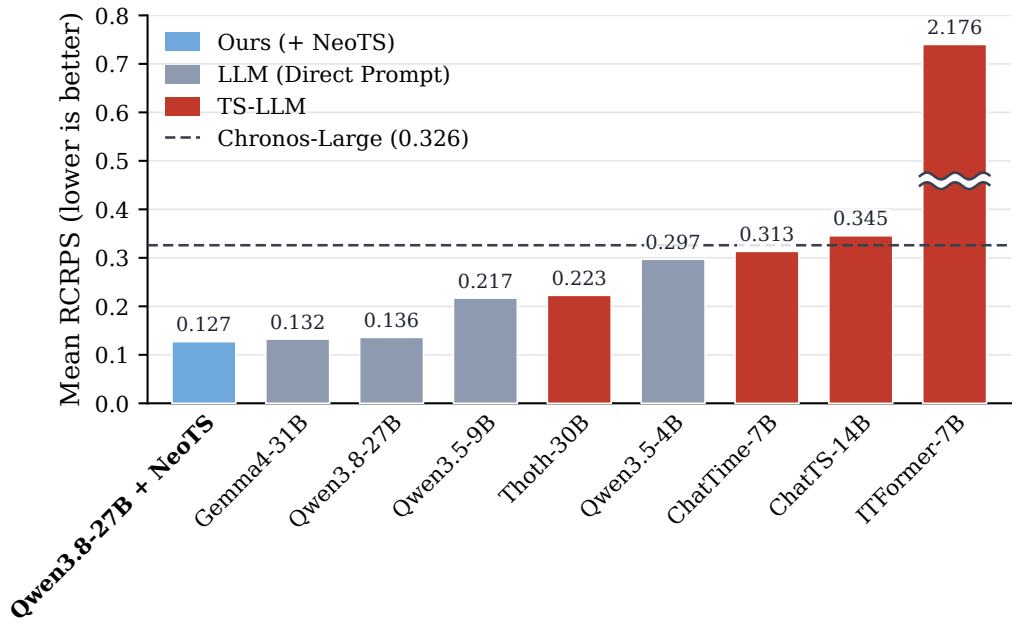

Figure 9: Mean RCRPS (lower is better) on Context is Key. General-purpose language models (gray) are prompted with the series written as decimal text, time-series language models (red) are trained on time-series, and Qwen3.8-27B+NEOTS (blue) keeps the backbone frozen and trains only the embeddings of the time-series words. The dashed line is Chronos-Large, a forecasting model that takes no text.

Table 20: Qwen3.5-4B on Context is Key under four training settings (RCRPS, lower is better). A dash marks a setting for which a share of the samples cannot be parsed, so no score can be computed.

Training		Input		
Backbone	TS words	Text	Block	RCRPS
fine-tuned	untrained	✓	×	–
fine-tuned	trained	✓	✓	–
frozen	untrained	✓	×	0.297
frozen	trained	✓	✓	0.267

when each input block is accompanied by a decimal list and as *block only* when it is not. In the setting of Section 4.2 the new words are learned from two signals. When a series appears as a block alone, the answer can be produced only by reading the block. When the same series appears both as a block and as decimal text, the two presentations share one answer, which aligns the block with the text. Table 21 gives the presentation of each family, together with the transcription task that is added in this setting, and Tables 23 and 27 list every task with its count.

Table 21: Presentation of CiT-Corpus in the setting of Section 4.2. "Block and text" shows each input series both as a block and as a decimal list, in either order. "Block only" shows the block alone. Transcription is added for this setting and is not part of Table 5.

Family	Block and text	Block only
forecasting, imputation, counterfactual, constrained	900,000	–
editing	300,000	–
arithmetic and read-out	400,000	400,000
multi-series relations	100,000	500,000
captioning	–	600,000
forecasting with covariates	–	200,000
transcription (added)	–	200,000
Total	1,700,000	1,900,000

G.2 Generation Procedure

Algorithm 1 gives the procedure for one example. Both sources share the last three stages and differ only in where the series comes from. All random draws are made from deterministically seeded per-partition generators, so the corpus is independent of the degree of parallelism used to build it.

Algorithm 1 Generation of one training example

Require: task T, seed σ

$\mathbf{x}, \mathbf{y}, \theta \leftarrow \textsc{Series}(T, \sigma)$ ▷ input series, target series, task parameters
  Chronos: window from the Chronos corpus (Appendix G.3)
  waveform: pattern + noise, rescaled by $A = 10^a$, $a \sim U(-2, 4)$ (Appendix G.4)
$k \leftarrow k(\mathbf{x})$ ▷ Section 4.2, shared by all blocks of the example
$I \leftarrow \textsc{Compose}(T, \theta, \sigma)$ ▷ opener + task sentence + word slots (Appendix G.5)
$P \leftarrow \textsc{Render}(I, \mathbf{x}, k)$ ▷ blocks (Equation 10), decimal list if T presents block and text
$Y \leftarrow \textsc{Format}(\mathbf{y}, k)$ ▷ text answer with $\max(0, 4 - k)$ decimals
assert Render and Format round-trip $\mathbf{x}$ and $\mathbf{y}$ at scale k
return (P, Y)

G.3 Chronos Windows

Sources. Series are drawn from the two components of the Chronos pre-training corpus (**?**), summarized in Table 22. Forty of its 220 partitions were selected uniformly across the corpus, preserving the 9:1 ratio of the two components. Each series contributes at most one window. Both components are distributed on the Hugging Face Hub under the Chronos datasets collection, whose constituent datasets carry their individual open licenses.

Window sampling. Series are assigned to a forecasting pool (3/4) or an imputation pool (1/4). Series shorter than 20 points are skipped.

- *Forecasting windows.* $L_{\text{in}} \sim U\{8, 128\}$, $L_{\text{out}} \sim U\{4, \min(48, \lfloor L_{\text{in}}/2 \rfloor)\}$, start position uniform. With probability 0.03 a short window is drawn instead ($L_{\text{in}} \sim U\{4, 15\}$, $L_{\text{out}} \sim U\{2, \max(2, \lfloor L_{\text{in}}/2 \rfloor)\}$), reflecting the short histories of context-aided benchmarks.
- *Imputation windows.* A window of length $L \sim U\{32, 128\}$ is placed uniformly inside the longest gap-free segment of the series. Masks are generated afterwards. Constant windows are rejected.

Table 22: Source series of the Chronos windows.

Component	Construction	Share
TSMixup	convex combinations (Dirichlet weights, up to three components, lengths 128–2,048) of mean-scaled series from the 28 Chronos training datasets	90%
KernelSynth	samples from Gaussian processes whose kernels are composed from a small bank (constant, white noise, linear, RBF, rational quadratic, periodic)	10%

- *Rejection rules.* Inputs with more than 20% missing values are rejected. A constant forecasting window with a constant target is retained with probability 0.5. If a forecasting target contains a missing value, the target is truncated before it, so that missing values occur in inputs only.

Tasks. Table 23 lists the six tasks with their construction rules, and Table 24 details the variants of the counterfactual and constrained tasks. The counterfactual and constrained tasks reuse the input of an existing example and replace only the instruction and the answer, so that the corpus contains pairs of examples with identical input but different instruction and answer. For the waveform variant of counterfactual forecasting, with $s = 10^k$, ℓ the last observed value, and H the horizon,

$$\begin{aligned} z_c &= \mathrm{clip}(\ell/s, -0.9, 0.9), \qquad P \sim U\{8, \min(32, \max(8, H))\}, \\ a &= \mathrm{round}_2\big(U(0.25, 0.85)\, s\, (1 - |z_c|)\big), \\ y_j &= z_c\, s + a\, w(j/P), \qquad j = 1, \ldots, H, \end{aligned} \tag{11}$$

where round_2 rounds to two significant digits and w is a unit sine, square, or triangle wave. The bound on z_c keeps $\mathbf{y}$ on the grid. For constrained forecasting, a bound is used only if a last-value forecast would violate it by more than 1.5% of s, so that the constraint is informative. It is placed at a margin $m \sim U(0.01, \min(0.08, g - 0.005))$ beyond the true extreme, where g is the distance between ℓ and the extreme in units of s, and is rounded to two decimals away from the truth.

Table 23: Tasks on Chronos windows. The answer of every task is a value list. Every example presents its input as block and text, with the text before or after the block with equal probability.

Task	Examples	Construction
forecasting	200,000	target: the L_{out} values after the window. Half of the windows with $L_{\mathrm{in}} \leq 64$ receive a date axis (15 min / hourly / daily / weekly, listed if $L_{\mathrm{in}} \leq 32$, compressed otherwise).
imputation, scattered mask	100,000	each point masked independently with probability $p \sim U(0.1, 0.5)$. Target: the masked values in order. Positions stated only if at most ten, and then with probability 0.5.
imputation, contiguous gap	100,000	one gap of length $G = \mathrm{clip}(\mathrm{round}(U(0.15, 0.5)\, L), 2, L - 4)$ removed. The two pieces are presented as two blocks. Target: the gap.
counterfactual forecasting	200,000	input of a forecasting example. The instruction defines the continuation (6 variants, Table 24).
counterfactual gap filling	100,000	input of a gap example. The instruction defines the fill (2 variants).
constrained forecasting	200,000	input and true future of a forecasting example. One sentence states a property of the future (5 variants). At most one variant per source example.

G.4 Generated Waveforms

Base patterns and noise. Series are synthesized on $t \in [0, 1]$, sampled at n points, from twelve base patterns (Table 25) with one of six noise models (Table 26), with baseline $B \sim U(-40, 40)$ and nominal amplitude 100 before rescaling. A constant baseline is never combined with Gaussian noise alone. The composite captioning task uses amplitude-modulated ($A(1 + \mu \sin(2\pi f_m t +$

Table 24: Variants of the counterfactual and constrained tasks. $\mathbf{x}$: input, $\ell = x_N$, $\mathbf{y}$: true future, H: horizon, G: gap length, x_{g_0-1} and x_{g_0+G}: values before and after the gap, $s = 10^k$, $\lceil\cdot\rceil_2$ and $\lfloor\cdot\rfloor_2$: rounding up and down to two decimals, r_2: rounding to two decimals.

Task	Variant	Answer or stated quantity
counterfactual forecasting	constant (last, min, max, zero)	ℓ, $\min \mathbf{x}$, $\max \mathbf{x}$, or 0, repeated H times
	repeat last	the last H points of $\mathbf{x}$ (cyclically if $N < H$)
	waveform	sine, square, or triangle wave of stated period P and amplitude a about ℓ, Equation 11
counterfactual gap filling	linear bridge	$\mathrm{linspace}(x_{g_0-1}, x_{g_0+G}, G+2)$ without endpoints
	hold edge	x_{g_0-1} repeated G times
constrained forecasting	upper bound	$\lceil \max \mathbf{y} + ms \rceil_2$, eligible if $(\ell - \max \mathbf{y})/s > 0.015$
	lower bound	$\lfloor \min \mathbf{y} - ms \rfloor_2$, eligible if $(\min \mathbf{y} - \ell)/s > 0.015$
	band	both bounds, eligible if either condition holds
	peak	$\mathrm{r}_2(\max \mathbf{y})$, eligible if $\lvert \max \mathbf{y} - \ell \rvert / s > 0.05$
	shift	$\mathrm{r}_2(\lvert \bar{\mathbf{y}} - \bar{\mathbf{x}} \rvert)$ and its direction, eligible if $\lvert \bar{\mathbf{y}} - \bar{\mathbf{x}} \rvert / s > 0.05$

$\phi_m)) \sin(2\pi f_c t + \phi_c) + B$, $\mu \in [0.5, 0.9]$) and damped ($Ae^{-\alpha t} \sin(2\pi f t + \phi) + B$, $\alpha \in [1.5, 4]$) oscillations instead. The transcription and covariate-forecasting tasks use a Gaussian random walk with probability 0.3.

Table 25: Base patterns ($t \in [0, 1]$, $B \sim U(-40, 40)$, $\mathrm{saw}(u) = 2(u - \lfloor u + \frac{1}{2} \rfloor)$, $\mathrm{tri}(u) = 2|\mathrm{saw}(u)| - 1$).

Pattern	$x(t)$	Parameters
constant	B	
linear increase / decrease	$\pm At + (B \mp A/2)$	$A \sim U(0.8, 1.2) \cdot 100$
exponential growth	$ae^{bt} + c$, $a = \frac{A}{e^b - 1}$, $c = B - a$	$b \sim U(3, 4.5)$, $A \sim U(0.9, 1.1) \cdot 100$
exponential decay	$ae^{bt} + c$, $a = \frac{A}{1 - e^b}$, $c = B - ae^b$	$b \sim U(-4.5, -3)$, $A \sim U(0.9, 1.1) \cdot 100$
sigmoid	$\frac{A}{1 + e^{-b(t-c)}} + (B - \frac{A}{2})$	$b \sim U(8, 14)$, $c \sim U(0.35, 0.65)$, $A \sim U(0.9, 1.1) \cdot 100$
Gaussian bump	$A \exp(-((t - b)/c)^2) + B$	$b \sim U(0.35, 0.65)$, $c \sim U(0.12, 0.22)$, $A \sim U(0.9, 1.1) \cdot 100$
level shift	$B \mp \frac{A}{2}$ $(t < p)$, $B \pm \frac{A}{2}$ $(t \geq p)$	$A \sim U(0.8, 1.2) \cdot 100$, $p \sim U(0.2, 0.8)$, sign uniform
sine	$A \sin(2\pi t/\lambda + \phi) + B$	$A \sim U(0.45, 0.55) \cdot 100$, $\lambda \sim U(0.15, 0.45)$, $\phi \sim U(0, 2\pi)$
square	$A \operatorname{sgn} \sin(2\pi t/\lambda + \phi) + B$	as sine
sawtooth	$A \, \mathrm{saw}(t/\lambda + \phi) + B$	as sine, $\phi \sim U(0, 1)$
triangle	$A \, \mathrm{tri}(t/\lambda + \phi) + B$	as sine, $\phi \sim U(0, 1)$

Table 26: Noise models applied to a base pattern x of length n ($\sigma_n = n/512$).

Noise	$\tilde{x}_i$	Parameters
clean	x_i	
smooth / noisy	$x_i + \varepsilon_i$, $\varepsilon_i \sim \mathcal{N}(0, \sigma^2)$	$\sigma = 0.8$ / 8
positive / negative spikes	$x_i + \varepsilon_i \pm \delta_i$, $\varepsilon_i \sim \mathcal{N}(0, 1.5^2)$	$\delta_i \sim U(20, 35)$ at $\max(2, \mathrm{round}(U\{4, 8\}\sigma_n))$ positions, 0 elsewhere
mixed spikes	both signs	$\max(2, \mathrm{round}(U\{3, 6\}\sigma_n))$ positions of each sign

Amplitude and scale. Each example is peak-normalized and multiplied by $A = 10^a$ with $a \sim U(-2, 4)$, drawn independently of the waveform. The exponent $k = \lceil a \rceil$ is therefore uniform on

$\{-1, \ldots, 4\}$ (realized shares 16.6–16.7%), and the within-decade peak $A/10^k$ is log-uniform on $(0.1, 1]$, matching the Benford-like distribution of the Chronos windows. All series of one example share A, since several tasks are decided by relative magnitude across blocks. Arithmetic constants are drawn on a grid of 10^{k-2} so that they are exactly representable at the answer precision 10^{k-4}.

Tasks. Table 27 lists the tasks with their sampling parameters. Additional constraints that make individual tasks well posed:

- *Alignment tasks* (lead/lag, sub-sequence localization). Uniqueness is verified on the final grid: every incorrect shift must differ from the correct one by at least 0.02 in z at some position, and the example is re-drawn otherwise.
- *Multi-series questions.* The distinguishing property must exceed a numeric margin before rescaling: a difference of means of at least 35, a ratio of variances of first differences of at least 3, or a net movement of at least 50 for the target and at most 20 for the others.
- *Constant addition and subtraction.* With probability 0.2 the answer is placed outside the input range $[-1, 1] \cdot 10^k$, so that the answer is not restricted to the input range.
- *Spike repair.* The jitter of the clean series is drawn once and kept identical in input and answer, and spikes are at least 2.5σ above it, so that the answer cannot be regenerated from its own prefix and only a point-wise reading of the input reduces the loss.
- *Forecasting with covariates.* A target series T is an affine function of a reference series R. In the known-future variant, the history of both series and the future of R are given and the answer is the future of T. In the leading-indicator variant, T lags R by ℓ steps, so the next $H \leq \ell$ values of T are determined by the observed values of R. Both series are presented as blocks.

G.5 Instructions

An instruction is composed of an optional opener (probability 0.35), a task sentence drawn from a per-task pool, and word slots (Table 28), yielding on the order of 10^5 surface forms per task (Honovich et al., 2023). The last two sentences of every pool (one for pools shorter than eight sentences) are never sampled during training. System prompts are drawn from a pool of 18. Whether the instruction precedes the block or refers back to it ("the series above") is decided per example with probability 0.5. Templates whose slots cannot be filled for a given example are filtered before sampling rather than degraded to a default sentence.

G.6 Verification

Every rendered example is checked automatically: the number of entries in a decimal list equals the number of value words of its block, the text answer round-trips to the stored series at the scale of the example, no block token falls inside the label span, and the realized exponent equals the intended one. Length and scale statistics of the rendered corpus are recomputed from the stored token ids and agree with the generation rules (Appendix G.7). The same rendering code is used at training and test time.

G.7 Statistics

Table 29 gives rendered lengths and answer tokens in the setting of Section 4.2. The training set has 3,600,000 examples and 1.63×10^9 tokens including prompts, with a maximum length of 1,752 tokens. Of the examples, 53% present their input as block only, all among the waveform tasks (70% of that subset, the remaining 30% block and text). The largest share of answer tokens held by a single task is 11% (transcription). For the Chronos windows the exponent follows the corpus: 82% of the windows have a maximum absolute value between 1 and 10 ($k = 1$), $k = 0$ for 12%, $k = 2$ for 5%, and the remaining 1% spread over $k \in \{-2, -1, 3, 4, 5, 6\}$, which is why the generated waveforms are needed to supervise the scale word at every exponent. For the waveforms the exponent is uniform over $\{-1, \ldots, 4\}$ by construction. Series lengths of the Chronos windows span 4–128 points for forecasting inputs (median 65) and 32–128 for imputation windows (median 78).

Table 27: Tasks on generated waveforms. Lengths are in points and $U\{a, b\}$ denotes a uniform integer draw. Transcription is the task added in Section 4.2.

Task	Answer	Examples	Length	Sampling
captioning				
waveform type	sentence	100,000	$U\{64, 448\}$	pattern and noise uniform
mean, std, min, max	sentence	100,000	$U\{64, 448\}$	non-constant series only
noise type	sentence	100,000	$U\{64, 448\}$	
pattern, noise, statistics	sentence	100,000	$U\{64, 448\}$	
composite oscillation	sentence	100,000	$U\{64, 448\}$	amplitude-modulated or damped, $p = 1/2$ each
periodicity, period length	sentence	100,000	$U\{64, 448\}$	periodic or aperiodic, $p = 1/2$ each
editing				
time reversal	value list	100,000	$U\{16, 64\}$	
spike repair, whole series	value list	100,000	$U\{16, 96\}$	
spike repair, index span	value list	100,000	$U\{16, 96\}$	span $U\{8, \min(24, n - 4)\}$
multi-series relations				
copy a designated series	value list	100,000	$U\{16, 64\}$ per block	$U\{2, 3\}$ blocks, target uniform
split: output both parts	two value lists	100,000	$U\{16, 48\}$ per part	boundary stated $p = 1/2$
lead/lag between two series	label or lag	100,000	$U\{32, 128\}$	lag $U\{2, 16\}$, direction or lag size asked $p = 1/2$
locate sub-sequence B in A	integer	100,000	A: $U\{48, 128\}$, B: $U\{8, 24\}$	uniqueness verified
continuation match	label or value list	100,000	context $U\{32, 128\}$, $H \sim U\{16, 48\}$	$U\{2, 4\}$ candidates, written out $p = 0.3$
multi-series question	label or value list	100,000	$U\{32, 128\}$ per block	$U\{2, 3\}$ blocks, 8 properties uniform, written out $p = 0.3$
arithmetic and read-out				
add / subtract a constant	value list	200,000	$U\{8, 48\}$	out-of-range answer $p = 0.2$
multiply by a factor	value list	100,000	$U\{8, 48\}$	factor $U(0.5, 5)$, $\lvert\text{factor} - 1\rvert \geq 0.1$, as factor or percentage
negate	value list	100,000	$U\{8, 48\}$	
pairwise add / subtract	value list	200,000	$U\{8, 64\}$	two blocks, shared scale
value read-out	number or date	100,000	$U\{8, 48\}$	6 question forms, index or date axis
first index above threshold	integer	100,000	$U\{8, 96\}$	$t \sim U(\text{lo} + 0.25r, \text{hi} - 0.1r)$, $r = \text{hi} - \text{lo}$, answer stable under rounding of t
forecasting with covariates				
known-future covariate, $T = aR + b$	value list	100,000	$L \sim U\{16, 40\}$, $H \sim U\{4, 8\}$	$a \sim \pm U(0.5, 2)$, $b \sim U(-50, 50)$
leading indicator, $T_t = aR_{t-\ell} + b$	value list	100,000	$\ell \sim U\{4, 12\}$, $H \leq \ell$	ℓ stated $p = 1/2$
added in Section 4.2				
transcription	value list	200,000	$U\{8, 64\}$	random walk $p = 0.3$
Total		2,700,000		

Table 28: Word slots and representative templates.

Slot	Values
{noun}	series, time series, sequence, signal, data series, history, measurement series
{unit}	points, values, steps, observations, entries
{Verb} (forecasting)	predict, forecast, project, extrapolate, estimate
{Fillverb} (imputation)	fill in, impute, reconstruct, recover, estimate
series labels	"Series 1", "A", "[1]"
Template	**Example**
forecasting	"{Verb} the next {n} {unit} of this {noun}."
imputation	"{Fillverb} every gap and output only the filled-in values, in position order."
constraint	"Keep in mind that the forecast window is limited to values between {lo} and {hi}."
multi-series question	"Identify the {noun} that {prop}. Respond with the label only."
threshold	"The {noun} crosses {t} at some point. Report the first index above {t}."

Table 29: Rendered sequence lengths (tokens, prompt and answer) and answer tokens. No example exceeds 1,752 tokens.

Series	Examples	Mean	p50	p99	Max	Answer tokens	Mean answer
Chronos windows	900,000	592.8	579	1,154	1,752	1.01×10^8	112
generated waveforms	2,700,000	405.4	365	1,169	1,712	3.51×10^8	130

G.8 Relation to CiT-Bench

The training series consist solely of the Chronos corpus and our own generated waveforms. The generator of the waveforms and the generators of the CiT-Bench tracks share no code, functions, or random seeds, and no benchmark series appears in the corpus. TSMixup series are convex combinations of source series and cannot be traced back individually, so this statement is made at the level of the named source datasets.

G.9 Examples

Figures 10 and 11 show examples rendered by the generation pipeline as presented to the model in the setting of Section 4.2. [block, N pts, k] stands for the $N + 3$ tokens of Equation 10, and long value lists are abbreviated.

H Transcription Curriculum

The 100,000 transcription examples that open the epoch are half of the 200,000 transcription examples in the training set, chosen at random. They are shuffled among themselves, and the remaining examples retain their order, so the multiset of examples and the number of steps are unchanged. The transcription loss settles near 0.55 after this segment and does not regress once transcription examples are diluted into the normal mixture.

Forecasting

```
system:  You analyse numerical sequences and answer questions about
them.
user:  Forecast how this history continues for 10 more entries.
Values:  0.0125 0.0006 -0.0112 -0.0228 ...  -0.3955 -0.3961
[block, 78 pts, k=0]
answer:  Here is the forecast:
-0.3968 -0.3974 -0.398 -0.3985 -0.399 -0.3996 -0.4001 -0.4005 -0.401
-0.4014
```

Imputation, scattered mask

```
user:  Estimate every gap and output only the filled-in values, in
position order.
[block, 121 pts, k=1]
Values:  1.156 0.585 NaN NaN NaN NaN -0.814 0.914 NaN -0.088 ...
-0.129 -0.209
answer:  Missing values:
-0.848 0.086 1.51 0.233 1.308 ...  0.267 -1.01
```

Constrained forecasting (band)

```
user:  [block, 8 pts, k=1]
Values:  -1.972 -0.884 -1.693 -0.961 0.304 3.236 -0.442 0.685
That is a series with 8 values.  Estimate how it continues over the
next 4 values.
Keep in mind that the forecast window is limited to values between
-1.75 and 0.6.
answer:  Here is the continuation:
-1.399 -0.062 0.098 -0.282
```

Figure 10: Examples on Chronos windows.

Waveform type

```
user:  The following data series comes from a measurement session.
Determine the pattern type of this data series.
[block, 309 pts, k=-1]
answer:  The series traces an exponential decay shape.
```

Lead/lag

```
user:  [1]
[block, 40 pts, k=2]
[2]
[block, 40 pts, k=2]
Between [1] and [2], which recording began first?
answer:  [1].
```

Transcription

```
user:  [block, 42 pts, k=0]
Output the steps of the series block as a plain numeric list.
answer:  Values:
-0.0853, -0.1542, -0.2087, -0.2419, ...  -0.2274, -0.1823
```

Add a constant (out-of-range answer)

```
user:  Output this sequence with 6200 added to every value.
Values:  7443 5183 3590 2517 1659 1041 745 411 91 45 45 -134 -77 -88
[block, 14 pts, k=4]
answer:  The resulting series:
13643 11383 9790 8717 7859 7241 6945 6611 6291 6245 6245 6066 6123
6112
```

First index above a threshold

```
user:  Index:  1, 2, 3, ..., 25
[block, 25 pts, k=2]
The sequence crosses 22.18 at some point.  Report the first index
above 22.18.  Your answer should be a single number.
answer:  1
```

Figure 11: Examples on generated waveforms.