LOTUS: LEARNING TO UNDERSTAND TIME-SERIES BY LEARNING TO FORECAST

Anonymous authors
Paper under double-blind review

ABSTRACT

Time-series language models typically align numerical signals with language using labels, captions, or question–answer pairs. Although such supervision encourages linguistic representations, it does not explicitly require them to preserve information about how the underlying time-series signal evolves. We introduce *forecast grounding*, a pretraining principle that requires language-aligned time-series representations to remain predictive of the future evolution before they are adapted to downstream tasks. We instantiate this principle in LOTUS, which divides each time-series channel into non-overlapping patches and maps them into the LLM embedding space. It then predicts the next patch from the LLM hidden states of the time-series tokens, encouraging the learned representation to retain temporal information. To capture both temporal changes and periodic patterns, we also propose a *temporal–spectral patch representation* that combines each raw time-series patch with its spectral representation before mapping it into the LLM embedding space. Across three language model backbones and six downstream evaluation datasets, LOTUS achieves the best average performance with minimal trainable parameters, reaching 59.91% accuracy and outperforming the strongest matched baseline by 3.36%. Our representation analyses further show that LOTUS uses the numerical input and preserves temporal structure in its learned patch representations. These results establish forecast grounding as a promising pretraining principle, enabling compact time-series language models to outperform models 25–50× their size on TSRBench.

1 INTRODUCTION

Time series describe processes that evolve over time, making both the observed values and their temporal dynamics important for understanding the underlying signal. Time-series foundation models exploit this structure through objectives such as forecasting and reconstruction to learn transferable numerical representations (Ansari et al., 2024; Das et al., 2024; Woo et al., 2024; Nie et al., 2023). At the same time, many practical tasks require understanding time series through natural language, such as identifying when an abnormal event occurred, comparing changes across variables, or answering questions about a physiological signal. Recent time-series language models address this setting by mapping numerical observations into representations that pretrained large language models (LLMs) can process (Li et al., 2025; Cheng et al., 2025; Xie et al., 2025; Langer et al., 2026). The effectiveness of this approach therefore depends on what information these time-series representations preserve.

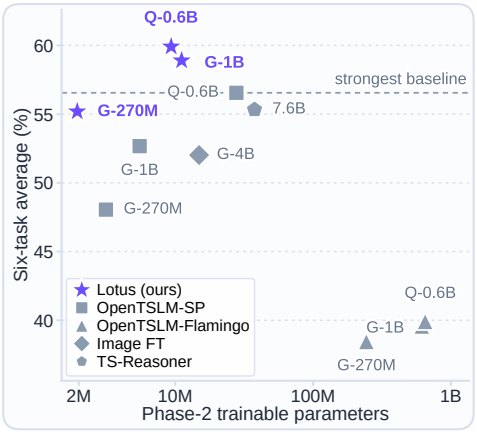


Figure 1: **Six-task average exact match accuracy versus trainable parameters (LoRA).** G and Q denote Gemma3 and Qwen3 backbones, respectively.

Existing time-series language models are primarily aligned through language supervision, including class labels, captions, instructions, and question–answer pairs (Langer et al., 2026; Xie et al., 2025).

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.

These objectives encourage the representation to preserve information needed to produce the correct language output, but they do not explicitly require it to retain how the underlying signal may evolve. For example, two motion segments may both correspond to *walking* while differing in speed, local dynamics, and likely continuation. The same language target may therefore correspond to temporal states with different future behavior. We argue that forecasting provides a natural additional constraint for learning such representations. Based on this idea, we introduce **forecast grounding**: during pretraining, a language-aligned time-series representation is also required to predict the future evolution of its source signal. If two observations share the same language target but lead to different futures, the forecasting objective provides an additional learning signal that distinguishes them. Forecasting therefore serves as a pretraining constraint that encourages the representation supplied to the LLM to preserve future-relevant temporal information, rather than as an additional downstream task.

We instantiate this principle in LOTUS. Each channel is divided into non-overlapping patches and mapped by an encoder and projector to soft tokens in the LLM embedding space. Temporal and channel positions identify the structure of multivariate inputs, while the causal LLM provides interaction across signal patches and text. During pretraining, language examples train the model to generate answers, while forecasting examples train it to predict the next patch from the LLM states of the signal tokens. As a result, the forecasting loss directly shapes the representation the language model uses. After pretraining, the model is jointly adapted to different downstream time-series tasks. Beyond the pretraining objective, we also consider how local time-series information is represented before entering the LLM. We propose a *temporal–spectral patch representation* that combines the raw temporal patch with its Fourier transform before mapping it into the LLM embedding space. The temporal view preserves local values and changes, while the spectral view exposes frequency content within each patch. Longer-range dependencies across patches are subsequently modeled by the language model. We evaluate LOTUS with three language backbones on six downstream splits covering activity recognition, sleep staging, ECG question answering, and general time-series question answering. As shown in Figure 1, LOTUS achieves the best average accuracy in every matched backbone group while using only a small number of trainable parameters.

Our contributions are threefold:

1. We propose **forecast grounding**, which uses future prediction during pretraining to encourage language-aligned time-series representations to preserve future-relevant temporal information.
2. We propose a **temporal–spectral patch representation** that combines the raw time-series patch with its spectral representation before mapping it into the LLM embedding space. This allows the model to capture both local temporal changes and frequency patterns.
3. We evaluate LOTUS with three language backbones on six downstream splits. LOTUS achieves the best average accuracy in every matched backbone group, demonstrating the effectiveness of combining forecast grounding with temporal–spectral patch representations for time-series language modeling.

2 RELATED WORK

2.1 TIME-SERIES LANGUAGE MODELS AND REASONING

Recent work extends pretrained language models to numerical time series and studies their ability to perform classification, interpretation, and reasoning over signals. SensorLLM aligns motion signals with textual trend descriptions before activity-recognition tuning, while InstructTime formulates time-series classification through multimodal language modeling (Li et al., 2025; Cheng et al., 2025). ChatTS pairs synthetic multivariate signals with attribute descriptions and instructions for reasoning over temporal attributes and relationships, while Time-MQA develops large-scale question–answer data for numerical analysis and open-ended interaction (Xie et al., 2025; Kong et al., 2025). In physiological domains, OpenTSLM studies activity recognition, sleep staging, and ECG question answering, while GEM grounds language generation in measurable ECG characteristics (Langer et al., 2026; Lan et al., 2025). TS-Reasoner instead incorporates representations from a pretrained time-series foundation model and aligns them with an LLM through instruction tuning (Yu et al., 2026b).

Despite this progress, reasoning directly over numerical signals remains difficult. Merrill et al. (2024) show that strong LLMs remain unreliable when time series are represented directly as text. Existing approaches address this problem mainly by introducing stronger language supervision, task-specific instruction data, or more informative numerical features. These directions improve what the language model is taught or what numerical information it receives, but place less emphasis on how the underlying time-series representation is shaped before downstream reasoning. LOTUS focuses on this representation. We encode each patch using both its temporal values and spectral information, providing the language model with complementary views of local signal behavior. We further use forecasting during pretraining to encourage the resulting representation to retain information about future signal evolution. The goal is therefore not to introduce another reasoning objective, but to learn a stronger numerical representation on which later language understanding and reasoning can operate.

2.2 TIME-SERIES–LANGUAGE ALIGNMENT

A central challenge in time-series language modeling is mapping continuous numerical signals into representations that pretrained language models can effectively process. Existing methods adopt several alignment mechanisms. Projection-based approaches map encoded time-series segments directly into the LLM embedding space (Li et al., 2025; Langer et al., 2026), while cross-attention architectures retain separate signal representations that language states can attend to (Langer et al., 2026; Lan et al., 2025). TS-Reasoner instead projects representations from a pretrained time-series foundation model into the language-model space (Yu et al., 2026b). These approaches establish effective connections between numerical signals and language, but the resulting representations are primarily shaped through language or downstream task supervision.

A related line of work uses the time series itself as a learning signal. PatchTST learns representations through masked-patch reconstruction (Nie et al., 2023), while SimMTM reconstructs the original series from multiple masked views to preserve temporal variation (Dong et al., 2023). Prediction has also been explored in language-model-based time-series methods. Time-LLM reprograms time-series patches using text prototypes before passing them to a frozen LLM for forecasting (Jin et al., 2024), while TEST aligns time-series embeddings with a text-prototype space for prediction with a frozen language model (Sun et al., 2024). More closely related to our predictive objective, aLLM4TS uses causal next-patch and multi-patch prediction to learn transferable time-series representations with an LLM (Bian et al., 2024). These methods demonstrate that reconstruction and prediction can provide useful supervision for learning temporal representations, but they are developed primarily for forecasting or conventional time-series tasks. LOTUS brings predictive supervision into time-series–language alignment for downstream language tasks through **forecast grounding**. During pretraining, we predict future patches from the post-LLM signal states, so the forecasting objective directly constrains the representations that are later used for language understanding and reasoning. Together with the temporal–spectral patch representation, this encourages the language model to operate on signal representations that capture both local temporal dynamics and frequency-domain structure while retaining information relevant to future signal evolution.

3 METHODOLOGY

Figure 2 summarizes LOTUS in three parts. First, the architecture divides each time-series channel into patches and maps each patch to a soft time-series token in the language-model embedding space. This design allows the representation of each patch to be contextualized jointly with other patches and language. We use *patch* to refer to a segment of the original time series and *soft time-series token* to refer to its projected representation supplied to the language model. Second, forecast grounding applies predictive supervision to the language-model states corresponding to these soft time-series tokens. We hypothesize that requiring the same representations used for language processing to retain information about future time series provides a complementary training signal for downstream time-series understanding. Third, downstream adaptation removes the next-patch decoder and continues to adapt the time-series encoder and LLM using the language objective alone.

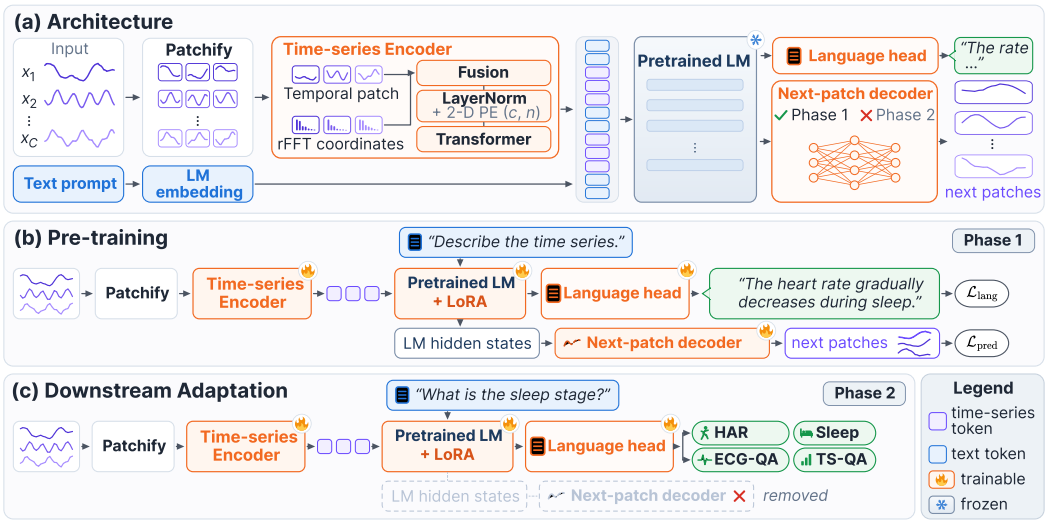


Figure 2: **Overview of LOTUS. (a) Architecture:** The architecture maps time-series patches to soft tokens that are processed jointly with text by the pretrained language model. **(b) Pretraining:** During forecast grounding pretraining, language supervision trains textual outputs while next-patch prediction constrains LLM's hidden states to retain future-relevant information. **(c) Downstream Adaptation:** During downstream adaptation, the next-patch decoder is removed and the model is trained with language supervision only.

3.1 LOTUS ARCHITECTURE

Patchification. Let a multivariate time series be $\mathbf{S} \in \mathbb{R}^{C \times L}$, where C is the number of channels and L is the sequence length. Each channel is normalized to zero mean and unit variance over its observed, non-padded extent, zero-padded to a multiple of the patch length P, and divided into non-overlapping patches

$$\mathbf{x}_{c,n} \in \mathbb{R}^P, \qquad c \in \{1, \dots, C\}, \quad n \in \{1, \dots, N\}, \tag{1}$$

where $N = \lceil L/P \rceil$. The input therefore yields CN patch positions. The encoder operates on each patch independently, while interaction across patches and channels is handled later by the pretrained language model. We represent each patch in both temporal and Fourier coordinates to expose complementary views of the same local segment. The temporal view preserves the raw within-patch evolution, while the Fourier view provides an explicit frequency-aware parameterization that may facilitate learning periodic structure.

Time-Series Encoder. While the raw temporal patch directly preserves local values and short-term changes, patterns may be less explicit in the time domain. We therefore propose a *temporal-spectral patch representation* which represents each patch using temporal and spectral views of its local dynamics. Specifically, for a patch $\mathbf{x}_{c,n}$, we use the raw patch as the temporal view. For the spectral view, let $\mathcal{F}(\mathbf{x}_{c,n}) \in \mathbb{C}^{P/2+1}$ denote its real FFT. We represent the complex spectrum by concatenating its real and imaginary components,

$$\mathbf{f}_{c,n} = [\mathrm{Re}\,\mathcal{F}(\mathbf{x}_{c,n}); \mathrm{Im}\,\mathcal{F}(\mathbf{x}_{c,n})] \in \mathbb{R}^{2(P/2+1)}. \tag{2}$$

The temporal and spectral views are embedded separately by learnable projections W_t and W_f, and fused by W_{fuse}

$$\mathbf{h}^t_{c,n} = W_t \mathbf{x}_{c,n}, \qquad \mathbf{h}^f_{c,n} = W_f \mathbf{f}_{c,n}, \tag{3}$$

$$\mathbf{h}_{c,n} = W_{\text{fuse}} \left[\mathbf{h}^t_{c,n}; \mathbf{h}^f_{c,n}\right]. \tag{4}$$

where $\mathbf{h}^t_{c,n} \in \mathbb{R}^{d_t}$ and $\mathbf{h}^f_{c,n} \in \mathbb{R}^{d_f}$ are the temporal and spectral representations of patch (c, n). Their concatenation is projected by W_{fuse} into a shared fused representation $\mathbf{h}_{c,n} \in \mathbb{R}^{d_h}$. The fused

representation is first normalized. To preserve the structure of the multivariate input, we add a two-dimensional positional encoding that identifies both the channel and temporal location of each patch. Specifically, we use a channel embedding $\mathbf{p}_c^{ch}$ and a patch-position embedding $\mathbf{p}_n^{pos}$, giving

$$\widetilde{\mathbf{h}}_{c,n} = \mathrm{LN}(\mathbf{h}_{c,n}) + \mathbf{p}_c^{ch} + \mathbf{p}_n^{pos}. \tag{5}$$

Here, LN denotes layer normalization. A transformer decoder then maps the patch representation to the language-model width d_{model}. We write the complete Time-series Encoder module in Figure 2(a), including the projector, as

$$\mathbf{z}_{c,n} = g_\phi(\mathbf{x}_{c,n}, c, n) \in \mathbb{R}^{d_{\mathrm{model}}}, \tag{6}$$

where ϕ denotes all trainable parameters of the time-series encoder. The vectors $\mathbf{z}_{c,n}$ are the CN soft time-series tokens shown in the figure.

Token Sequence. The encoded time series are inserted into the language-model input as soft tokens while preserving the original ordering of the multimodal input. Each time series $\mathbf{S}$ is mapped by $g_\phi(\mathbf{S})$ to a sequence of soft tokens and enclosed by learned $\langle ts \rangle$ and $\langle /ts \rangle$ delimiters. When an input contains multiple time series, each series is represented by its own delimited soft-token sequence, with any intervening text retained between them. Thus, text, captions, and time-series representations are interleaved in the same order as they appear in the original example. The target answer A is appended during teacher-forced training. The resulting multimodal token sequence, denoted by $\mathbf{E}$, is processed by the pretrained causal language model,

$$\mathbf{H} = LLM_\Theta(\mathbf{E}), \tag{7}$$

where $\mathbf{H}$ denotes the output hidden representations for all positions in the sequence.

3.2 Phase 1: Forecast Grounding Pretraining

Forecast grounding pretraining uses a mixture of forecasting, captioning, and time-series question-answering data, as shown in Figure 2(b). The time-series encoder and projector are trained jointly with the next-patch decoder, while the language model is adapted through LoRA. We denote the encoder parameters by ϕ, the next-patch decoder parameters by ψ, and the LoRA parameters by η.

Language Objective. For language generation tasks like captioning and question-answering examples, the supervision is a standard autoregressive language objective. Let $\mathcal{A}$ denote the target answer positions, and a_t denote the target token at position t. We optimize:

$$\mathcal{L}_{\mathrm{lang}} = -\sum_{t \in \mathcal{A}} \log p_{\phi,\eta}\left(a_t \mid \mathbf{E}_{<t}\right). \tag{8}$$

This objective jointly updates the time-series encoder's parameters ϕ and the LoRA parameters η through the language-model computation.

Forecasting Objective. Forecasting examples provide the predictive component of pretraining. For patch $\mathbf{x}_{c,n}$, let $u(c,n)$ denote the corresponding time series token position. The next-patch decoder d_ψ predicts the normalized succeeding patch from the final hidden states of the LLM:

$$\widehat{\mathbf{x}}_{c,n+1} = d_\psi\big(\mathbf{H}_{u(c,1)}, \ldots, \mathbf{H}_{u(c,n)}\big)_n. \tag{9}$$

In implementation, d_ψ projects the language-model hidden states back to the time-series representation space and applies a causally masked Transformer decoder, followed by a linear prediction head that outputs the P values of the succeeding patch. The causal Transformer allows the prediction at position n to condition on $\mathbf{H}_{u(c,n)}$ as well as preceding states from the same channel.

Let $\mathcal{V} = \{(c,n) : 1 \le c \le C,\ 1 \le n < N\}$ denote the set of positions with an observed successor patch in the same channel. Sequence-padding patches and positions without a successor are excluded. The forecasting loss is

$$\mathcal{L}_{\mathrm{pred}} = \frac{1}{|\mathcal{V}|} \sum_{(c,n) \in \mathcal{V}} \|\widehat{\mathbf{x}}_{c,n+1} - \mathbf{x}_{c,n+1}\|_2^2. \tag{10}$$

The decoder therefore operates on the LLM's hidden state. Gradients from $\mathcal{L}_{\mathrm{pred}}$ propagate through the language-model computation and jointly update the time-series encoder's parameters ϕ, the LoRA parameters η, and the next-patch decoder's parameters ψ.

Optimization. We train on separate forecasting and language batches, so each optimization step uses either $\mathcal{L}_{\text{lang}}$ or $\mathcal{L}_{\text{pred}}$. Let D_{lang} and D_{pred} denote the language and forecasting data distributions, respectively. The overall pretraining objective is

$$\mathcal{L}_{\text{PG}} = \mathbb{E}_{b\sim\mathcal{D}_{\text{lang}}}[\mathcal{L}_{\text{lang}}(b)] + \mathbb{E}_{b\sim\mathcal{D}_{\text{pred}}}[\mathcal{L}_{\text{pred}}(b)]. \tag{11}$$

Because the Phase-1 corpora differ greatly in size, size-proportional sampling would cause the largest forecasting corpus to dominate training. We therefore upsample smaller sources and downsample larger ones according to predefined mixture weights, making the language–forecasting balance independent of corpus size. Language batches update ϕ and η through Eq. 8; forecasting batches update ϕ, η, and ψ through Eq. 10.

3.3 Phase 2: Downstream Adaptation

After pretraining, we remove the next-patch decoder and jointly adapt one model across downstream tasks, fully fine-tuning the encoder and projector while applying LoRA to the LLM:

$$\min_{\phi,\eta} \mathbb{E}_{b\sim\mathcal{D}_{\text{down}}}[\mathcal{L}_{\text{lang}}(b)]. \tag{12}$$

We balance the five corpora by sampling $B = 40{,}000$ examples from each per epoch. For a corpus of size N_k, we repeat the full corpus $q_k = \lfloor B/N_k \rfloor$ times, then fill the remaining $r_k = B - q_k N_k$ slots with examples selected without replacement using a seeded shuffle. When $N_k > B$, no full repetition is needed; we select a fixed subset of B examples and reuse it across epochs. The resulting $200{,}000$-example mixture gives each corpus equal representation, preventing larger datasets from dominating the sample count (more details in Appendix F).

4 Experiments

4.1 Datasets and Implementation Details

Phase 1. Our pretraining phase uses four datasets: half-hourly electricity demand from Monash (Godahewa et al., 2021), our synthetic multivariate corpus (details in Appendix E), TSQA multiple-choice questions on trend, volatility, seasonality, and outliers (Kong et al., 2025), and M4 series with generated captions (Makridakis et al., 2020). We retain OpenTSLM's language pretraining data (Langer et al., 2026), adding Monash and our synthetic as forecasting pretraining datasets.

Phase 2. The downstream adaptation phase finetunes the checkpoints of Phase 1 on five datasets: HAR (Langer et al., 2026), PAMAP2 (Reiss & Stricker, 2012), Sleep-EDF (Kemp et al., 2000), ECG-QA (Oh et al., 2023; Wagner et al., 2020), and TimeMQA (Kong et al., 2025), whose MCQ and T/F splits are evaluated separately. Unless otherwise stated, we finetune our method and all baselines on those datasets jointly (as mentioned in Section 3.3). Each corpus receives a budget of $40{,}000$ examples so that none dominates the mixture. Details on phase 2 datasets are in Appendix F.

Training and Evaluation Protocols. For training, during Phase 1, the encoder, projector, decoder, and LLM LoRA adapters are trained jointly. During Phase 2, the next-patch decoder is removed, and the time-series interface and LLM LoRA adapters are trained using the language objective alone. Appendix A.1 provides the complete optimization and LoRA configurations. For evaluation, every baseline is evaluated under greedy decoding: the model writes a rationale and an explicit answer, which is parsed and compared with the reference. We report classification accuracy on HAR, PAMAP2, Sleep-EDF, and TimeMQA, and generative accuracy under normalization on ECG-QA. Appendix A.2 details answer normalization and the held-out benchmarks.

4.2 Baselines

We include a set of well-known baselines with different model sizes, backbones and input modalities in **Reference**, including fine-tuned Gemma3-4B (Gemma Team, 2025) and fine-tuned TS-Reasoner (Yu et al., 2026b). We then match backbones and sizes across **Text-tokenized fine-tuning**, using

Table 1: **Accuracy on downstream tasks.** Avg. is the mean over the six splits; # Params counts trainable parameters in downstream adaptation. Per backbone block, **bold** is best and underline second best per column; † marks OpenTSLM's original evaluation protocol (Langer et al., 2026).

	Baseline	# Params	HAR	Sleep	ECG	PAMAP2	MCQ	T/F	**Avg.**
Reference	Chance	—	12.50	14.29	0.98	8.33	25.00	50.00	18.52
	Llama-3.1-8B-Instruct, zero-shot	0	9.26	24.52	18.36	9.83	8.74	9.42	13.35
	Qwen2.5-7B-Instruct, zero-shot	0	15.97	31.40	31.76	9.40	10.02	3.34	16.98
	Mistral-7B-Instruct-v0.3, zero-shot	0	7.07	21.40	21.31	6.84	24.31	24.62	17.59
	Gemma3-4B-pt (image), zero-shot	0	1.97	0.00	0.00	0.00	17.48	31.61	8.51
	Gemma3-4B-pt (image), FT	14.9M	51.31	61.18	30.94	51.71	46.10	70.85	52.02
	TS-Reasoner-7B, zero-shot	0	17.26	29.89	13.84	15.81	36.25	0.00	18.84
	TS-Reasoner-7B, FT	37.6M	59.62	60.65	30.09	58.80	54.39	68.42	55.33
Gemma3-270M	Text-tokenized, zero-shot	0	0.00	0.00	0.00	0.00	15.57	22.80	6.40
	Text-tokenized, FT	1.9M	**64.00**	64.14	23.42	47.44	38.28	72.81	51.68
	OpenTSLM-SP	3.1M	54.38	65.12	9.66	52.14	38.59	68.39	48.05
	↳ *sequential*†	3.1M	1.47	0.00	2.56	0.00	41.36	70.52	19.32
	OpenTSLM-Flamingo	317M	62.55	68.01	30.10	54.70	12.37	3.04	38.46
	↳ *sequential*†	317M	3.77	0.00	0.00	9.40	20.04	4.86	6.34
	LOTUS	1.9M	58.38	**70.54**	**33.16**	**55.15**	**39.41**	**74.47**	**55.19**
Gemma3-1B	Text-tokenized, zero-shot	0	0.00	0.00	3.46	0.00	5.76	10.35	3.26
	Text-tokenized, FT	6.5M	57.08	70.65	35.24	51.71	44.70	78.12	56.25
	OpenTSLM-SP	5.5M	55.46	72.58	21.66	47.44	41.36	77.51	52.67
	↳ *sequential*†	15.3M	0.00	0.00	3.68	0.00	42.22	76.90	20.47
	OpenTSLM-Flamingo	1.3B	60.01	66.99	32.48	51.71	19.62	6.69	39.58
	↳ *sequential*†	1.3B	0.15	0.00	1.88	1.28	20.90	2.43	4.44
	LOTUS	11.1M	**62.67**	**77.10**	**37.66**	**52.56**	**44.98**	**78.42**	**58.90**
Qwen3-0.6B	Text-tokenized, zero-shot	0	5.86	1.51	13.24	4.70	36.03	16.11	12.91
	Text-tokenized, FT	5.0M	51.40	34.84	29.66	30.34	52.88	70.52	44.94
	OpenTSLM-SP	27.8M	63.25	70.54	**32.20**	47.44	54.64	71.20	56.55
	↳ *sequential*†	27.8M	4.00	30.54	5.02	3.42	48.19	72.37	27.26
	OpenTSLM-Flamingo	650M	60.54	45.27	28.12	46.58	37.13	21.88	39.92
	↳ *sequential*†	650M	26.51	40.97	4.86	14.53	31.13	18.24	22.71
	LOTUS	9.3M	**63.66**	**77.20**	31.58	**53.42**	**56.72**	**76.90**	**59.91**

digit strings and LoRA (Gruver et al., 2023); **OpenTSLM-SP**, using soft prompts; and **OpenTSLM-Flamingo**, using gated cross-attention (Langer et al., 2026). Each uses Gemma3-270M, Gemma3-1B (Gemma Team, 2025), and Qwen3-0.6B (Yang et al., 2025), sharing LOTUS's Phase 1 source corpora with method-specific supervision and Phase 2 downstream datasets. OpenTSLM rows marked with *sequential*† follow the original sequential fine-tuning recipe (Langer et al., 2026).

4.3 RESULTS

Main Results. We present the main results in Table 1. LOTUS achieves the best six-task average among the compared methods at each backbone size, and performs best in every dataset except HAR at the smallest Gemma model and ECG-QA on Qwen3. We achieve these results with far fewer trainable parameters than other baselines: on Qwen3-0.6B we train a third of what OpenTSLM-SP does and roughly a seventieth of what OpenTSLM-Flamingo does. Our smallest model, LOTUS on Gemma-270M comes within 0.14% of a fully fine-tuned TS-Reasoner-7B system (55.19% vs. 55.33%) whose backbone and trainable parameters are an order of magnitude larger. Overall, LOTUS on Qwen3-0.6B achieves the best average result of 59.91%, 4.58% above TS-Reasoner-7B-FT, while being **14 times smaller in backbone size**.

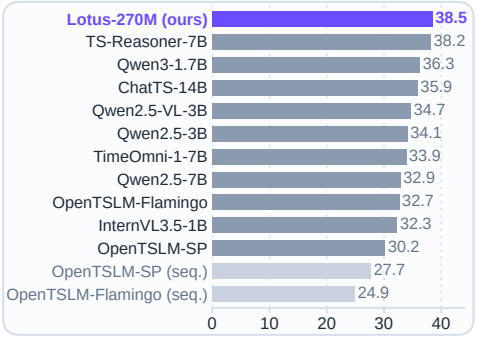
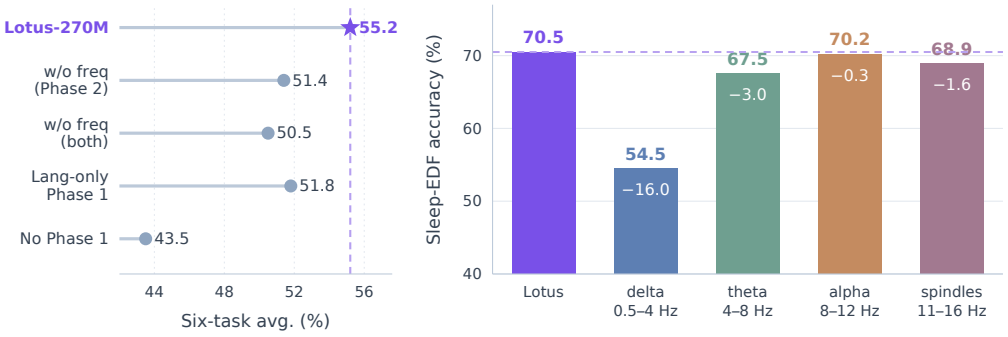


Figure 3: **TSRBench overall accuracy**. Per-subset results are in Appendix's Table 4.



(a) Component ablation.

(b) Frequency sensitivity.

Figure 4: **Component and input analyses on Gemma3-270M. (a)** Six-task average under ablations of pretraining and the frequency branch. **(b)** Sleep-EDF accuracy after removing individual frequency bands from the test signal. In-bar labels show percentage-point drops.

Sequential adaptation substantially degrades earlier-task performance. The *sequential*† baselines in Table 1 follow OpenTSLM's original sequential fine-tuning recipe and dataset order (HAR → Sleep → ECG), with PAMAP2 and TimeMQA appended. Compared with balanced joint-mixture training, sequential adaptation lowers average accuracy by 17.21%–35.14%. Performance declines mainly on datasets encountered earlier, including eight settings with 0% accuracy. In contrast, accuracy on TimeMQA's MCQ and T/F splits, trained last, is largely preserved and sometimes matches the jointly trained counterparts. This pattern suggests that adapting to later tasks can compromise performance on earlier ones. We therefore use joint adaptation for the main comparison, exposing the shared encoder and projector to all datasets throughout training.

Zero-shot Results On TSRBench. We evaluate zero-shot reasoning ability on TSRBench (Yu et al., 2026a), using our Gemma3-270M checkpoint. As Figure 3 shows, **LOTUS-270M scores highest among the thirteen baselines**, ahead of **backbones 25–50× larger** like TS-Reasoner-7B and ChatTS-14B. The per-subset results in Table 4. LOTUS achieves the highest overall TSRBench accuracy among the evaluated models, with strong performance in event prediction and comparative analysis. From the results, we believe LOTUS transferable strength shows most in reasoning from signal evidence to a conclusion. Detailed per-task results are reported in Appendix C.

5 ANALYSIS

5.1 COMPONENT ABLATION

Figure 4a compares LOTUS with variants that remove pretraining or frequency information. *No Phase 1* fine-tunes directly in Phase 2 with a randomly initialized encoder and projector, lowering average accuracy from 55.2% to 43.5%. *Lang-only Phase 1* uses TSQA and M4 at the same pretraining step budget and scores 3.4% below full LOTUS on average. Removing the frequency branch in Phase 2 alone or both phases lowers performance by 3.8% and 4.7% relative to full LOTUS, respectively, with the largest losses on ECG-QA and PAMAP2. In these variants, each time-series patch is embedded using only the temporal projection $W_t\mathbf{x}_{c,n}$.

5.2 FREQUENCY ANALYSIS

Figure 4b removes one frequency band at a time from the Sleep-EDF test signal. Suppressing delta (0.5–4 Hz) lowers accuracy by 16.0%, 3.0% for theta (4–8 Hz), 1.6% for spindles (11–16 Hz), and 0.3% for alpha (8–12 Hz). The largest drop follows delta suppression, indicating that the trained model is particularly sensitive to this low-frequency content in Sleep-EDF. Appendix B provides complementary evidence of signal dependence: shuffling patches, replacing the signal with noise, or zeroing it reduces HAR accuracy to 26.9%, 26.5% and 32.2%, respectively. These results show

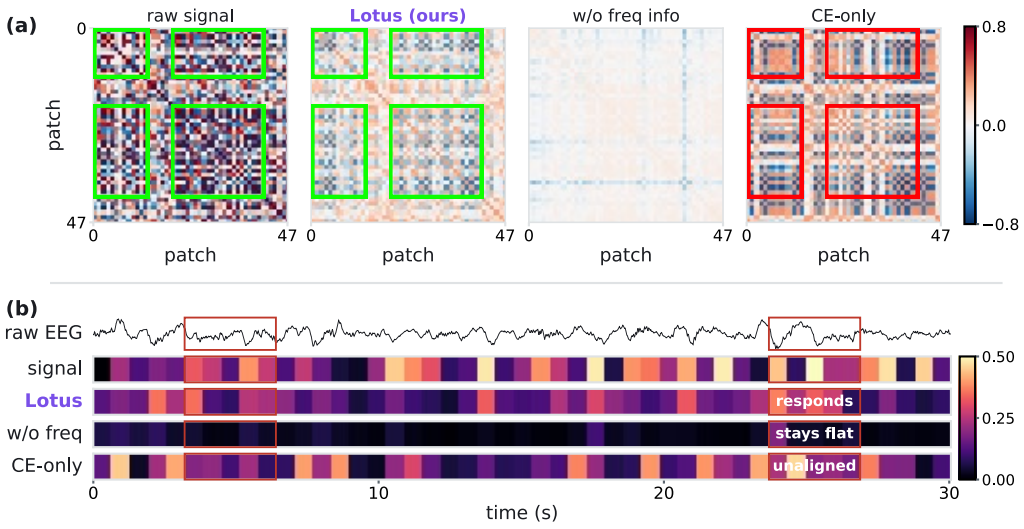


Figure 5: **Representation analysis. (a)** Pairwise patch similarities. **Green** boxes highlight patterns shared by LOTUS and the raw signal; **red** boxes mark corresponding regions where these patterns are less apparent with cross-entropy (CE)-only training. **(b)** Patch colors indicate cosine distance from the window-average representation. Boxes highlight spans selected for comparison.

that signal information contributes to predictions beyond the accompanying text. The frequency-branch ablations examine how explicitly providing Fourier coordinates affects learning: removing the branch lowers six-task average accuracy by 3.8% in Phase 2 alone and 4.7% in both phases.

5.3 REPRESENTATION ANALYSIS

We compare patch-pair similarities in raw signals and learned representations over held-out Sleep-EDF windows of 47 patches. For patch vectors u_i and u_j, we compute cosine similarity as $s_{ij} = u_i^\top u_j / (\|u_i\|_2 \|u_j\|_2)$ and centre each panel on its off-diagonal mean. **Figure 5a** uses patch indices on both axes, with blocks identifying groups of similar patches. LOTUS (second panel) reproduces prominent raw-signal blocks, highlighted by **green** boxes, suggesting that patches with similar signal content retain similar representations. Without the frequency branch, the third panel shows a more uniform pattern with less visible correspondence to these structures. The cross-entropy (CE)-only representation retains varied patterns (fourth panel), but those marked by **red** boxes align less closely with the raw signal. **Figure 5b** shows a Sleep-EDF EEG signal and patch-wise cosine distances $d_i = 1 - u_i^\top \bar{u} / (\|u_i\|_2 \|\bar{u}\|_2)$ over time in seconds, where $\bar{u} = \frac{1}{N} \sum_{j=1}^{N} u_j$ is the window-average representation. Brighter colours indicate greater deviation from the window average; the raw-signal row is separately rescaled. In the highlighted spans (red boxes) for comparison, LOTUS varies alongside the signal, whereas the no-frequency variant remains relatively flat and CE-only changes align less closely. Together, these findings suggest that LOTUS preserves both relationships between patches and local variation within the signal.

6 CONCLUSION

We introduce LOTUS, a time-series language model built on *forecast grounding*: pretraining language-aligned representations to retain information about future signal evolution. LOTUS combines next-patch prediction from LLM hidden states with a temporal–spectral patch representation. Across three language-model backbones and six downstream splits, LOTUS achieves the best average performance among matched baselines. Ablations show gains from forecast grounding and frequency information, while representation analyses suggest that predictive supervision helps preserve temporal structure. These findings support forecasting as a pretraining objective for time-series language models, even when downstream tasks require only language outputs.

AI USE STATEMENT

Large language models were not involved in the design of our approach. We used them solely to polish the writing of this paper (grammar and style). All methods, experiments, and analyses were designed and run by the authors, who take responsibility for the content.

ETHICS STATEMENT

This work uses existing time-series datasets and synthetic data. Physiological and activity signals raise privacy concerns, and model predictions may reflect dataset biases or produce misleading explanations. LOTUS is evaluated as a research system; its results do not establish suitability for clinical decision-making or other high-stakes applications.

REPRODUCIBILITY STATEMENT

We describe the architecture and training objectives in Section 3 and provide training configurations and evaluation procedures in Appendix A. Appendices E and F document data construction, splits, and sampling, while Appendices H and I report computational resources and prompt templates to support reproduction of our experiments.

REFERENCES

Abdul Fatir Ansari, Lorenzo Stella, Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Sundar Rangapuram, Sebastian Pineda Arango, Shubham Kapoor, Jasper Zschiegner, Danielle C. Maddix, Hao Wang, Michael W. Mahoney, Kari Torkkola, Andrew Gordon Wilson, Michael Bohlke-Schneider, and Yuyang Wang. Chronos: Learning the language of time series. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=gerNCVqqtR.

Abdul Fatir Ansari, Oleksandr Shchur, Jaris Küken, Andreas Auer, Boran Han, Pedro Mercado, Syama Sundar Rangapuram, Huibin Shen, Lorenzo Stella, Xiyuan Zhang, Mononito Goswami, Shubham Kapoor, Danielle C. Maddix, Pablo Guerron, Tony Hu, Junming Yin, Nick Erickson, Prateek Mutalik Desai, Hao Wang, Huzefa Rangwala, George Karypis, Yuyang Wang, and Michael Bohlke-Schneider. Chronos-2: From univariate to universal forecasting, 2025.

Yuxuan Bian, Xuan Ju, Jiangtong Li, Zhijian Xu, Dawei Cheng, and Qiang Xu. Multi-patch prediction: Adapting language models for time series representation learning. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, 2024. URL https://arxiv.org/abs/2402.04852.

Mingyue Cheng, Yiheng Chen, Qi Liu, Zhiding Liu, Yucong Luo, and Enhong Chen. Instructime: Advancing time series classification with multimodal language modeling. In *Proceedings of the Eighteenth ACM International Conference on Web Search and Data Mining*, pp. 792–800, 2025.

Abhimanyu Das, Weihao Kong, Rajat Sen, and Yichen Zhou. A decoder-only foundation model for time-series forecasting. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, volume 235 of *PMLR*, pp. 10148–10167, 2024. URL https://proceedings.mlr.press/v235/das24c.html.

Jiaxiang Dong, Haixu Wu, Haoran Zhang, Li Zhang, Jianmin Wang, and Mingsheng Long. SimMTM: A simple pre-training framework for masked time-series modeling. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. URL https://arxiv.org/abs/2302.00861.

Gemma Team. Gemma 3 technical report, 2025. URL https://arxiv.org/abs/2503.19786.

Rakshitha Godahewa, Christoph Bergmeir, Geoffrey I. Webb, Rob J. Hyndman, and Pablo Montero-Manso. Monash time series forecasting archive. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1, 2021.

Nate Gruver, Marc Finzi, Shikai Qiu, and Andrew Gordon Wilson. Large language models are zero-shot time series forecasters. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pp. 19622–19635, 2023.

Ming Jin, Shiyu Wang, Lintao Ma, Zhixuan Chu, James Y. Zhang, Xiaoming Shi, Pin-Yu Chen, Yuxuan Liang, Yuan-Fang Li, Shirui Pan, and Qingsong Wen. Time-LLM: Time series forecasting by reprogramming large language models. In *International Conference on Learning Representations (ICLR)*, 2024.

Bob Kemp, Aeilko H. Zwinderman, Bert Tuk, Hilbert A. C. Kamphuisen, and Josefien J. L. Oberye. Analysis of a sleep-dependent neuronal feedback loop: The slow-wave microcontinuity of the EEG. *IEEE Transactions on Biomedical Engineering*, 47(9):1185–1194, 2000.

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. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.acl-long.1437. URL https://aclanthology.org/2025.acl-long.1437/.

Xiang Lan, Feng Wu, Kai He, Qinghao Zhao, Shenda Hong, and Mengling Feng. GEM: Empowering MLLM for grounded ECG understanding with time series and images. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=idtZwmjakN.

Patrick Langer, Thomas Kaar, Max Rosenblattl, Maxwell A. Xu, Winnie Chow, Martin Maritsch, Robert Jakob, Ning Wang, Juncheng Liu, Aradhana Verma, Brian Han, Daniel Seung Kim, Henry Chubb, Scott Ceresnak, Aydin Zahedivash, Alexander Tarlochan Singh Sandhu, Fatima Rodriguez, Daniel McDuff, Elgar Fleisch, Oliver Aalami, Filipe Barata, and Paul Schmiedmayer. OpenTSLM: Time-series language models for reasoning over multivariate medical text- and time-series data. In *Proceedings of the 43rd International Conference on Machine Learning (ICML)*, 2026.

Zechen Li, Shohreh Deldari, Linyao Chen, Hao Xue, and Flora D. Salim. SensorLLM: Aligning large language models with motion sensors for human activity recognition. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 354–379. Association for Computational Linguistics, 2025. URL https://aclanthology.org/2025.emnlp-main.19/.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. The M4 competition: 100,000 time series and 61 forecasting methods. *International Journal of Forecasting*, 36(1): 54–74, 2020.

Mike A. Merrill, Mingtian Tan, Vinayak Gupta, Thomas Hartvigsen, and Tim Althoff. Language models still struggle to zero-shot reason about time series. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 3512–3533, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.201. URL https://aclanthology.org/2024.findings-emnlp.201/.

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

Jungwoo Oh, Gyubok Lee, Seongsu Bae, Joon-myoung Kwon, and Edward Choi. ECG-QA: A comprehensive question answering dataset combined with electrocardiogram. In *Advances in Neural Information Processing Systems*, volume 36, pp. 66277–66288. Curran Associates, Inc., 2023.

Attila Reiss and Didier Stricker. Introducing a new benchmarked dataset for activity monitoring. In *2012 16th International Symposium on Wearable Computers (ISWC)*, pp. 108–109. IEEE, 2012.

Chenxi Sun, Hongyan Li, Yaliang Li, and Shenda Hong. TEST: Text prototype aligned embedding to activate LLM's ability for time series. In *International Conference on Learning Representations (ICLR)*, 2024.

Ege Onur Taga, Muhammed Emrullah Ildiz, and Samet Oymak. TimePFN: Effective multivariate time series forecasting with synthetic data. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, 2025.

Patrick Wagner, Nils Strodthoff, Ralf-Dieter Bousseljot, Dieter Kreiseler, Fatima I. Lunze, Wojciech Samek, and Tobias Schaeffter. PTB-XL, a large publicly available electrocardiography dataset. *Scientific Data*, 7(1):154, 2020.

Wenxuan Wang, Kai Wu, Yujian Betterest Li, Dan Wang, and Xiaoyu Zhang. Synthetic series-symbol data generation for time series foundation models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=xB1ZNgq0Xp.

Gerald Woo, Chenghao Liu, Akshat Kumar, Caiming Xiong, Silvio Savarese, and Doyen Sahoo. Unified training of universal time series forecasting transformers. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, volume 235 of *PMLR*, pp. 53140–53164, 2024. URL https://proceedings.mlr.press/v235/woo24a.html.

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. *Proceedings of the VLDB Endowment*, 18(8):2385–2398, 2025.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Fangxu Yu, Xingang Guo, Lingzhi Yuan, Haoqiang Kang, Hongyu Zhao, Lianhui Qin, Furong Huang, Bin Hu, and Tianyi Zhou. TSRBench: A comprehensive multi-task multi-modal time series reasoning benchmark for generalist models. In *Forty-third International Conference on Machine Learning*, 2026a. URL https://openreview.net/forum?id=S9G3yu6vHl.

Fangxu Yu, Hongyu Zhao, and Tianyi Zhou. TS-Reasoner: Aligning time series foundation models with LLM reasoning. *Transactions on Machine Learning Research*, 2026b. URL https://arxiv.org/abs/2510.03519.

APPENDIX CONTENTS FOR
Lotus: Learning to Understand Time-series by Learning to Forecast

A Experiment Details **14**

A.1 Training Configuration 14

A.2 Evaluation Details 14

A.3 Training Details 14

A.4 Full Training Data Ablation Results 14

B Input Diagnostics **15**

C Per-task TSRBench Results **15**

D Qualitative Comparisons **15**

E Phase 1 Training Data **16**

E.1 Overview 16

E.2 Synthetic Multivariate Corpus 16

E.3 Dataset Statistics 22

F Phase-2 Training Data **22**

F.1 Balanced Mixture. 23

F.2 Evaluation Subsets. 23

G Computational Cost **23**

G.1 Reading A Series 23

G.2 Training on A Series 23

H Computational Resources **24**

H.1 Hardware and Precision 24

H.2 Training Time 24

H.3 Vocabulary-dependent Memory 25

H.4 Configurations Across Backbones 25

I Prompt Templates **25**

I.1 Phase 1 25

I.2 Phase 2 25

I.3 TSRBench Evaluation 30

A Experiment Details

A.1 Training Configuration

Table 2 summarizes the recipe shared across the reported runs. Phase 1 uses source-homogeneous batches, so each update applies either the language or prediction loss. Phase 2 draws the same fixed example budget from each corpus and applies only the language loss.

A.2 Evaluation Details

All matched systems use greedy decoding. Each response contains a rationale followed by an explicit answer field. The parser extracts that field, applies case folding and whitespace normalization, and maps dataset-specific textual variants to the task's canonical label. An output is counted as incorrect when no label can be recovered. ECG-QA uses the released answer normalization. The headline metric is generative accuracy, followed by an unweighted mean across the four non-overlapping datasets; it is not a token-level language-model metric.

A.3 Training Details

We provide the detailed configs for both phases in Table 2.

Table 2: Configuration of the two phases. Settings shared by both phases are listed first.

	Phase 1	Phase 2
Shared		
Patch size	$P = 32$, non-overlapping	
Encoder	Transformer decoder, width 64, FF 256, output 768; 95,680 params	
Position encoding	fixed 2-D sinusoidal over (patch index, channel index); not learned	
Next-patch Decoder	linear projection to the encoder width, causally masked Transformer decoder, linear head to Poutputs;	
Optimiser	AdamW (Loshchilov & Hutter, 2019), weight decay 10^{-2}	
LR (enc./proj.)	1×10^{-4}	
LR schedule	linear warmup over the first 3% of steps, then linear decay to 0	
Grad. clipping	global norm 1.0	
Precision	AMP fp16 with `GradScaler`	
Hardware	8× NVIDIA V100 (2 nodes × 4); effective batch 128	
Phase-specific		
Data	forecasting, QA, captioning	six CoT corpora
Mixture	weighted 1:2:1:1 (60% num. / 40% ling.)	balanced, 40,000 examples per task per epoch
Epoch size	task-weighted sampler	$5 \times 40{,}000 = 200{,}000$
Objective	$\mathcal{L}_{\text{CE}} + \mathcal{L}_{\text{MSE}}$	$\mathcal{L}_{\text{CE}}$ only
LoRA	r=8, α=16, all attention and MLP projections	r=8, α=16, all attention and MLP projections (excluding Decoder's parameters)
LR (LoRA)	2×10^{-4}	2×10^{-4}
Batch size	16	1
Grad. accumulation	1	16
Selection	validation loss	mean generative EM, 192 validation items per task
Early stopping	patience 5	per task, patience 8

A.4 Full Training Data Ablation Results

In this section, we provide the detailed results for removing elements from our training data. Table 3 gives the per-task results behind Figure 4a. Language-only pretraining lowers every signal task (HAR -4.0, Sleep -4.1, ECG-QA -2.3, PAMAP2 -6.4) while leaving MCQ unchanged. Removing the frequency branch is felt most on ECG-QA (-7.4 in Phase 2 only, -7.9 in both phases) and

PAMAP2 ($-13.7/-10.7$); HAR is the one task that improves without it ($+3.2$ in the Phase-2-only variant).

Table 3: **Ablations on Gemma3-270M, per task.** Each variant changes one ingredient and is otherwise trained and evaluated as in Table 1. **Bold** is best per column.

Variant	HAR	Sleep	ECG	PAMAP2	MCQ	T/F	Avg.
LOTUS	58.38	**70.54**	**33.16**	**55.15**	**39.41**	**74.47**	**55.19**
w/o frequency branch (Phase 2 only)	**61.55**	69.46	25.80	41.45	37.10	72.95	51.39
w/o frequency branch (both phases)	55.89	67.54	25.24	44.44	37.31	72.64	50.51
Language-only Phase 1 (same budget)	54.38	66.45	30.84	48.72	39.23	71.16	51.79

B INPUT DIAGNOSTICS

We test whether predictions can be recovered from the text prompt or label prior alone. Figure 6 evaluates the Gemma3-270M LOTUS checkpoint of Table 1 on the full test splits (8,222 HAR and 930 Sleep-EDF items) after shuffling the signal, replacing it with Gaussian noise, or setting it to zero. HAR accuracy falls from 58.4% to 26.5–32.2%, bracketing its empirical majority-class rate of 28.3%; Sleep-EDF falls from 70.5% to 45.2% under every corruption, its majority-class rate being 45.3%. Thus, the model's predictions depend materially on the numerical input.

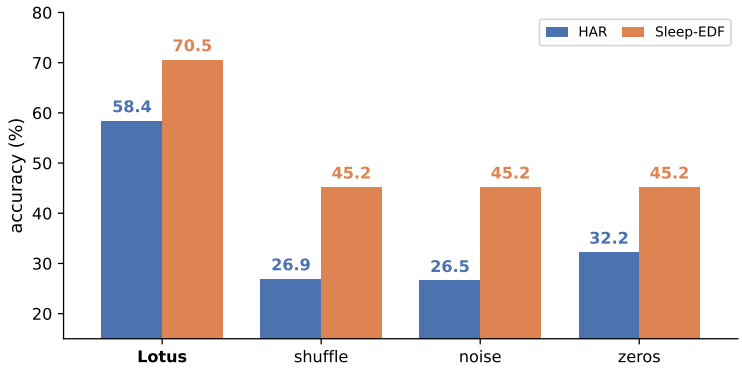


Figure 6: **Input diagnostics.** Accuracy after corrupting the numerical input, on the Gemma3-270M checkpoint and the full test splits (HAR n=8,222, Sleep-EDF n=930).

C PER-TASK TSRBENCH RESULTS

Figure 3 reports the aggregate; the full per-subset breakdown behind it is given in Table 4.

Evaluation protocol. We follow the benchmark's evaluation (Yu et al., 2026a) and use its released scorer unmodified: it extracts the option letter from the generated text, drops responses from which no letter can be recovered, and micro-averages accuracy over subsets.

D QUALITATIVE COMPARISONS

In this section, we provide qualitative comparisons between LOTUS and the baselines on Sleep-EDF, HAR, ECG-QA, and PAMAP2. Figures 7–10 each show one test item with every system built on Gemma3-270M. Each figure shows the input series as the encoder receives it, the question (instructions omitted for clarity; see Appendix I), and each model's rationale up to its first answer; any text generated after that answer is truncated. Highlights were checked against measurements on the input, reported under each series: red marks a wrong answer or a claim the measurements contradict, and green marks the correct answer and the claim it rests on. Claims the measurements support are left unmarked, including in wrong answers.

Table 4: **Per-task TSRBench accuracy (%) over 14 subsets**. Columns are grouped into perception, reasoning, prediction, and decision-making.

	Perception				Reasoning						Prediction		Decision		
Model	PR	NU	AD	CA	ER	CD	AR	TR	NR	IR	TSF	EP	QualDM	QuantDM	Overall
Qwen2.5-7B	40.3	50.0	43.3	64.0	17.7	28.2	32.0	20.0	32.1	34.3	31.8	42.0	32.6	19.4	32.9
Qwen3-1.7B	39.1	51.2	33.9	48.5	16.0	35.3	42.7	28.1	28.0	32.0	40.5	69.4	32.5	17.7	36.3
Qwen2.5-3B	40.7	40.7	33.1	46.9	25.4	20.1	24.0	28.0	33.3	20.2	32.4	66.9	34.0	20.0	34.1
Qwen2.5-VL-3B	56.1	56.0	45.0	53.1	28.0	17.1	26.7	28.8	30.0	32.0	26.5	66.7	33.4	23.7	34.7
InternVL3.5-1B	47.0	40.2	34.9	48.2	23.1	27.7	24.7	27.5	27.0	25.0	39.6	27.6	32.2	20.3	32.3
TimeOmni-1-7B	42.8	49.3	37.3	53.6	24.5	26.1	32.0	21.0	21.7	28.8	30.0	74.7	28.2	22.7	33.9
TS-Reasoner-7B	48.6	52.9	38.9	57.1	22.3	25.0	54.0	23.9	36.5	22.4	28.1	83.8	33.1	23.3	38.2
ChatTS-14B	46.6	48.3	41.4	57.7	24.3	37.0	57.3	21.9	42.1	33.0	29.8	25.8	27.6	43.7	35.9
OpenTSLM-SP	41.5	43.7	38.0	42.5	28.1	27.8	32.0	27.2	28.8	27.0	21.0	49.0	23.6	18.0	30.2
↳ *sequential*†	40.5	48.3	34.4	48.2	29.5	25.2	26.7	26.6	26.0	31.0	26.0	16.1	24.8	19.0	27.7
OpenTSLM-Flamingo	36.6	30.8	33.3	44.0	27.4	36.4	25.6	15.4	36.4	25.6	23.9	45.4	25.7	19.1	32.7
↳ *sequential*†	29.6	35.1	30.0	24.5	32.4	21.1	22.7	12.0	22.1	29.4	16.7	34.2	22.9	12.4	24.9
LOTUS	26.2	50.0	29.0	60.9	27.6	25.7	25.0	19.2	34.2	29.9	17.6	83.5	25.8	23.7	38.5

The examples show two recurring failure modes. Some baselines describe the signal correctly but answer a different question, as on ECG-QA, where two baselines report a q wave or ST depression that is present in the recording instead of the queried T-wave finding. Others misread the signal itself, as when OpenTSLM-SP describes the PAMAP2 window as stationary despite three large excursions. In all four examples, OpenTSLM-SP and the tokenized baseline also keep generating after their first answer, repeating tokens or restating the answer.

E PHASE 1 TRAINING DATA

E.1 OVERVIEW

Phase 1 draws source-homogeneous batches from four corpora with fixed sampling weights 1:2:1:1 (Section 4.1). The two forecasting sources therefore account for 60% of optimisation steps and the two language sources for the remaining 40%. These proportions are determined by the sampling weights rather than by corpus size.

Two corpora provide next-patch supervision. The first contains half-hourly electricity demand for five Australian states from the Monash archive (Godahewa et al., 2021). We split each series chronologically into the first 80% for training, the next 10% for validation, and the final 10% for testing. The second is the synthetic corpus described below. Because each row is generated independently rather than extracted from a shared history, we randomly partition complete rows in the same $80/10/10$ ratio; the same generated series never appears in more than one split. The two remaining corpora provide language supervision: 38.4K multiple-choice items concerning trend, volatility, seasonality, and outliers, and 80K M4 series (Makridakis et al., 2020) paired with generated captions.

E.2 SYNTHETIC MULTIVARIATE CORPUS

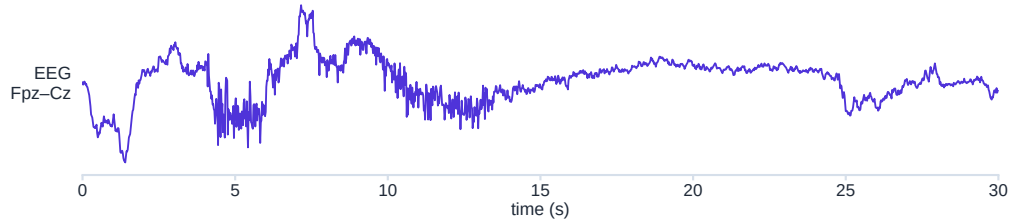
To obtain large-scale multivariate supervision without training on the downstream evaluation series, we construct a synthetic corpus of $844{,}275$ series. It combines four generator families (Table 5), none of which contributes more than 37% of the data. Two families impose cross-channel structure explicitly, KernelSynth contributes univariate diversity, and SymTime contributes series generated from symbolic processes.

KernelSynth. The univariate generator introduced by Chronos (Ansari et al., 2024) randomly composes a Gaussian-process (GP) covariance. For each series, it draws $j \sim \mathcal{U}\{1, \ldots, 5\}$ kernels with replacement from a bank of 33 configurations. The bank contains 21 `ExpSineSquared` configurations spanning several seasonal scales, together with parameterised `DotProduct`, `RBF`,

Sleep-EDF: sleep staging gold answer: **Non-REM stage 1**

Single-channel EEG (Fpz–Cz), 30 s at 50 Hz · test item 352 · all systems on Gemma3-270M

Input, as the encoder receives it (per-channel z-scores)



Measured on this input: the delta band (0.5–4 Hz) holds 45% of 0.3–30 Hz power, below the lower quartile of N3 (69%) and N4 (79%), so the slow waves of deep sleep are absent; separately, 80% of non-DC power lies below 0.3 Hz (slow drift).

Question

You are given a 30-second EEG time series segment. Your task is to classify the sleep stage based on analysis of the data. *[reasoning and format instructions omitted]* Possible sleep stages are: Wake, Non-REM stage 1, Non-REM stage 2, Non-REM stage 3, Non-REM stage 4, REM sleep.

OpenTSLM-SP answered **Wake** ✗

The time-series data presented exhibits characteristics that are indicative of a particular sleep stage. The signal shows relatively high-frequency fluctuations and a notable amount of variability throughout the interval. These features suggest a level of cortical activity that is consistent with a more alert state. The presence of frequent, rapid changes in amplitude and the absence of large, slow waves typically associated with deeper sleep stages further support this interpretation. The overall pattern aligns with the characteristics expected during a state of wakefulness, where the brain exhibits more dynamic and less synchronized activity compared to the deep, restorative sleep stages. **Answer: Wake** *[… continues "No distinct distinct distinct …", 'distinct' ×80]*

OpenTSLM-Flamingo answered **Wake** ✗

The time-series data presented exhibits characteristics that are indicative of a particular sleep stage. The signal shows relatively high-frequency fluctuations and a notable amount of variability throughout the interval. These features suggest a level of neural activity that is consistent with a more alert state. The presence of rapid changes and the overall irregularity in the amplitude of the signal are not typical of deep sleep stages, where the EEG would display slower, more synchronized waveforms. Instead, the observed pattern aligns with a state of heightened awareness and responsiveness. **Answer: Wake**

Text-tokenized FT answered **Wake** ✗

The observed EEG signal displays characteristics that are typically associated with a more active and alert state. The signal shows fluctuations and variability that suggest higher frequency and irregularity, which are indicative of a more wakeful or responsive condition. The presence of rapid changes and the lack of sustained low-frequency activity further support this interpretation. These features are consistent with a state where the brain is actively processing information or responding to external stimuli. Considering these and the other indications of the signal wakefulness, the observed sleep stage is identified as a more pronounced and active pattern. **Answer: Wake** *[… continues, emitting 'Answer: Wake' twice more]*

Lotus (ours) answered **Non-REM stage 1** ✓

The time series data presented shows a pattern that is characteristic of a specific sleep stage. The signal demonstrates relatively low amplitude fluctuations with some variability, which is typical of a lighter sleep stage where brain activity is more variable and less synchronized compared to deeper sleep stages. The presence of these fluctuations suggests a transition from wakefulness to a more stable sleep state, rather than the high amplitude, low-frequency waves associated with deeper sleep. The observed pattern aligns with the characteristics of a lighter sleep stage, where the brain is not fully at rest but is still responsive to external stimuli. **Answer: Non-REM stage 1.**

step that leads to the wrong answer, or a claim the measurements contradict

supported claim or step that leads to the correct answer **Bold**: final answer (red wrong, green correct)

[…] output after the first answer, truncated. Claims were checked against the measurements above; steps against the gold answer.

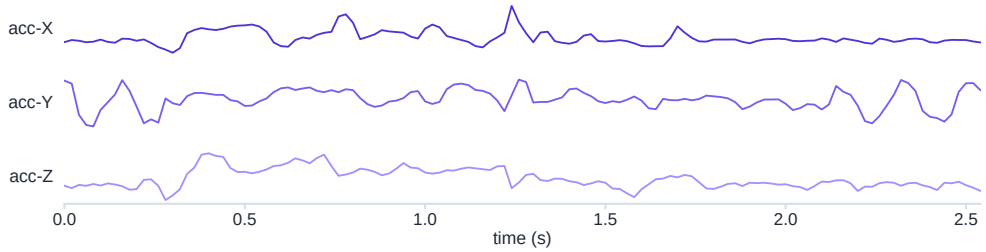
Figure 7: **Qualitative example, Sleep-EDF**. All three baselines read the epoch as wakefulness; LOTUS places it in a lighter sleep stage, consistent with its spectral centroid.

HAR: activity recognition

gold answer: **sitting**

Three-axis accelerometer, 2.56 s at 50 Hz · test item 5576 · all systems on Gemma3-270M

Input, as the encoder receives it (per-channel z-scores)



Measured on this input: mean absolute first difference 0.44, inside the interquartile range of only two of the eight classes, sitting and standing; on the Z axis a single 0.39 Hz swing carries 44% of the AC power and the 1–3 Hz step band only 10%, so there is no step cadence.

Question

You are given accelerometer data in all three dimensions. Your task is to classify the activity based on analysis of the data. *[reasoning and format instructions omitted]* Possible activity labels are: biking, lying, running, sitting, standing, walking, walking_down, walking_up.

OpenTSLM-SP — answered **standing** ✗

The accelerometer data over the 2.56 second window shows relatively low variability and consistent patterns across the X, Y, and Z axes. The X-axis data fluctuates within a narrow range, suggesting minimal lateral movement. Similarly, the Y-axis data maintains a steady pattern with slight variations, indicating limited vertical motion. The Z-axis also shows minor fluctuations, which implies that there is no significant forward or backward movement. The overall stability and lack of pronounced peaks or troughs across all axes suggest a lack of dynamic activity typically associated with more vigorous movements. Given these observations, the data is consistent with a scenario where the subject is largely stationary, with only minor adjustments in position or posture. **Answer: standing.** *[… continues, restating 'standing' twice]*

OpenTSLM-Flamingo — answered **walking** ✗

The accelerometer data over the 2.56 second window shows significant variability across all three axes, indicating substantial movement. The X-axis exhibits frequent fluctuations with both positive and negative peaks, suggesting lateral or forward-backward motion. The Y-axis also displays notable changes, with several peaks and troughs, indicating vertical movement or shifts in posture. The Z-axis shows a consistent pattern of oscillations, which could correspond to the up-and-down motion typically associated with steps. The presence of these dynamic and rhythmic patterns across all axes suggests a repetitive and active movement rather than a static position. Given the observed intensity and variability in the accelerometer readings, the activity is best classified as walking. **Answer: walking.**

Text-tokenized FT — answered **biking** ✗

The accelerometer data over the 2.56 second window shows significant variability across all three axes, indicating substantial movement. The X-axis exhibits frequent peaks and troughs, suggesting lateral or side-to-side motion. The Y-axis also displays fluctuations, though with less amplitude compared to the X-axis, which might indicate vertical movement such as bouncing or stepping. The Z-axis shows notable changes, with both positive and negative values, suggesting forward or backward motion. The presence of these dynamic patterns, along with the range of acceleration values, points towards an activity involving continuous movement and changes in position. Such characteristics are more aligned with an activity that involves active engagement and movement, rather than a stationary or minimal movement state. **Answer: biking.** *[… continues "Quiz: Answer: biking.Answer: biking. …"]*

Lotus (ours) — answered **sitting** ✓

The accelerometer data over the 2.56 second window shows relatively low variability and consistent patterns across the X, Y, and Z axes. The X-axis data fluctuates within a narrow range, suggesting minimal lateral movement. Similarly, the Y-axis data maintains a steady pattern with slight variations, indicating limited vertical motion. The Z-axis also shows a stable trend with minor fluctuations, which implies a lack of significant forward or backward movement. These characteristics collectively suggest a lack of dynamic activity, as one would expect more pronounced and frequent changes in acceleration if there were active movement or changes in position. The overall stability and low intensity of the signals across all axes are more indicative of a stationary or low-movement activity. **Answer: sitting.**

step that leads to the wrong answer, or a claim the measurements contradict

supported claim or step that leads to the correct answer **Bold**: final answer (red wrong, green correct)

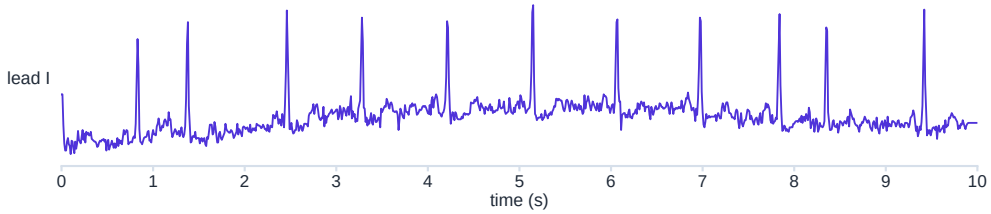
[…] output after the first answer, truncated. Claims were checked against the measurements above; steps against the gold answer.

Figure 8: **Qualitative example, HAR**. LOTUS and OpenTSLM-SP share their first 360 characters verbatim and differ only in the final label; OpenTSLM-Flamingo describes a stepping motion that the Z-axis spectrum does not show.

ECG-QA: question answering over a 12-lead ECG gold answer: **low amplitude t-wave**

Lead I shown (the queried lead), 10 s at 100 Hz · test item 438 · all systems on Gemma3-270M

Input, as the encoder receives it (per-channel z-scores)



Measured on this input (lead I): T-wave amplitude 0.10 mV, lower than in 9 of the 11 other leads, and |T|/R = 0.23. A small, normal septal q wave (|Q|/R = 0.14) and slight ST depression (–0.05 mV) are also present.

Question

You are an expert cardiologist analyzing an ECG (electrocardiogram). Clinical Context: 74-year-old male patient. 12-lead ECG. clinical recording. recorded with AT-6 C 5.5. Signal quality: baseline drift noted, static noise present, burst noise present, electrode artifacts present. extra beats detected during recording. pacemaker present. **Question: What form-related symptoms does this ECG show in lead I?** *[reasoning and format instructions omitted]* Options: digitalis effect, high qrs voltage, inverted t-waves, low amplitude t-wave, low qrs voltages in the frontal and horizontal leads, non-diagnostic t abnormalities, non-specific st changes, non-specific st depression, non-specific st elevation, non-specific t-wave changes, none, q waves present, t-wave abnormality, voltage criteria (qrs) for left ventricular hypertrophy.

OpenTSLM-SP answered **none** ✗

In analyzing the ECG, we focus on lead I to determine the form-related symptoms. The presence of baseline drift, static noise, burst noise, and electrode artifacts can complicate interpretation, but careful examination reveals the underlying cardiac patterns. The rhythm appears regular, and the pacemaker spikes are visible, indicating pacemaker activity. In lead I, the T-wave morphology is crucial for identifying any abnormalities. The T-wave in lead I does not show significant deviations that would suggest a specific pathology, such as inversion or flattening, which would be more indicative of a specific condition. The T-wave in lead I appears to maintain a normal amplitude, suggesting that there are no significant abnormalities present. Therefore, the ECG in lead I does not show any form-related symptoms. **Answer: none** *[… continues with an HTML fragment and a second 'Answer: none']*

OpenTSLM-Flamingo answered **q waves present** ✗

q waves are present in lead I, which can indicate previous myocardial infarction or other cardiac conditions. These waves are typically small and precede the QRS complex, suggesting a previous myocardial infarction. The presence of baseline drift, static noise, burst noise, and electrode artifacts can complicate the interpretation, but the focus remains on the morphology of the QRS complex in lead I. In this context, the presence of q waves in lead I is a significant finding, as it can be associated with the clinical symptoms of being in a patient with a pacemaker, which can alter the typical ECG findings. **Answer: q waves present.**

Text-tokenized FT answered **non-specific st depression** ✗

In analyzing the ECG, we focus on lead I to determine the form-related symptoms present. The presence of baseline drift, static noise, burst noise, and electrode artifacts can complicate interpretation, but careful examination reveals the underlying patterns. In lead I, the ST segment is slightly depressed, which is indicative of a non-specific ST depression. This finding is consistent with non-specific changes that can occur due to various factors, including the presence of a pacemaker, which can alter the typical ECG morphology. The ST depression in lead I is not pronounced enough to be considered diagnostic, but rather represents a subtle alteration in the ST segment. Given the clinical context and the observed ECG patterns, the form-related symptom in lead I is best described as non-specific ST depression. **Answer: non-specific st depression.** *[… continues after the answer]*

Lotus (ours) answered **low amplitude t-wave** ✓

In lead I, the T-wave morphology is crucial for determining the form-related symptoms. The T-wave in lead I appears to have a reduced amplitude compared to the other leads, which is indicative of low amplitude T-waves. This finding is consistent with the clinical context of a 74-year-old male with a pacemaker, where low amplitude T-waves can be a common finding. The presence of baseline drift, static noise, burst noise, and electrode artifacts can complicate the interpretation, but the consistent observation of low amplitude T-waves across the ECG leads, despite the noise and artifacts, supports the conclusion of low amplitude T-waves. **Answer: low amplitude t-waves.**

step that leads to the wrong answer, or a claim the measurements contradict

supported claim or step that leads to the correct answer **Bold**: final answer (red wrong, green correct)

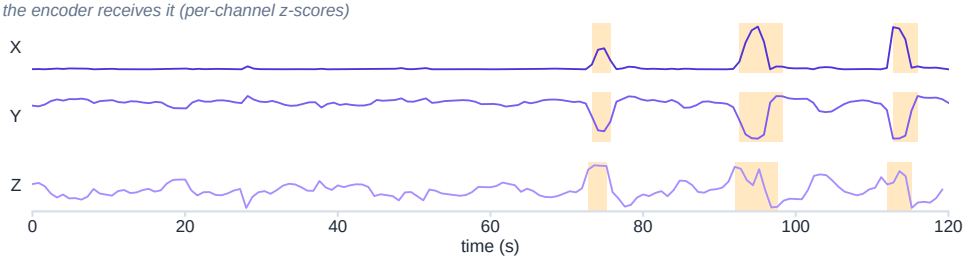
[…] output after the first answer, truncated. Claims were checked against the measurements above; steps against the gold answer.

Figure 9: **Qualitative example, ECG-QA**. The q wave and ST depression cited by two baselines are present in the recording but are not the queried finding; LOTUS identifies the reduced T-wave amplitude in lead I.

PAMAP2: activity recognition gold answer: **descending stairs**

Three-axis accelerometer, 150-point summary of a 120 s window · test item 50 · all systems on Gemma3-270M

Input, as the encoder receives it (per-channel z-scores)



Measured on this input: large excursions on all three axes at 72.8–75.2 s, 92.0–97.6 s and 112.0–115.2 s (shaded). The input has 1.25 samples per second, so step cadence is not resolved.

Question

You are given accelerometer data in all three dimensions. Your task is to classify the activity based on analysis of the data. *[reasoning and format instructions omitted]* Possible activity labels are: lying, sitting, standing, walking, running, cycling, nordic walking, watching TV, computer work, car driving, ascending stairs, descending stairs, vacuum cleaning, ironing, folding laundry, house cleaning, playing soccer, rope jumping.

OpenTSLM-SP answered **sitting** ✗

The accelerometer data over the two-minute window shows relatively low variability and consistent patterns across the X, Y, and Z axes. The X-axis data remains mostly stable with minor fluctuations, suggesting minimal lateral movement. The Y-axis exhibits some variation but lacks the pronounced peaks and troughs that would indicate significant vertical motion. The Z-axis also shows limited change, with a slight increase and decrease in values, but overall remains stable. These characteristics suggest low-intensity movement with occasional minor adjustments, which aligns with a stationary activity. **Answer: sitting.**
[… continues "array.array.array. …", 'array.' ×44]

OpenTSLM-Flamingo answered **ascending stairs** ✗

The time-series data from the accelerometer over a two-minute window reveals distinct patterns across the X, Y, and Z axes. Initially, there is a period of relative stability with minor fluctuations, followed by a series of pronounced peaks and troughs, particularly noticeable in the X and Z axes, suggesting intermittent bursts of movement. These peaks are indicative of dynamic activity, and the intensity and regularity of these fluctuations imply a repetitive motion that involves changes in elevation and direction. The Y-axis shows a significant spike towards the end, which aligns with a sudden movement that could be associated with a lifting or stepping motion. The overall pattern of the data suggests a rhythmic and repetitive activity with bursts of intensity, consistent with the physical exertion and movement associated with climbing. **Answer: ascending stairs.**

Text-tokenized FT answered **ascending stairs** ✗

The accelerometer data over the two-minute window shows distinct patterns across the X, Y, and Z axes. The X-axis data reveals occasional sharp peaks, suggesting intermittent bursts of movement or impact, which are consistent with changes in direction or changes in posture. The Y-axis displays a more consistent pattern, with fluctuations that indicate regular vertical movement, which is characteristic of an activity involving a rhythmic upward motion. The Z-axis also shows a regular pattern of peaks and troughs, indicating consistent vertical acceleration that aligns with the notion of repeated upward steps. The combination of these patterns, particularly the regularity and intensity observed in the X and Y axes, suggests a high level of consistent and repetitive movement. The presence of these bursts of acceleration in all three axes, particularly the X and Y axes, indicates an activity characterized by rhythmic and continuous motion. **Answer: ascending stairs.** *[… continues after the answer]*

Lotus (ours) answered **descending stairs** ✓

The time-series data from the accelerometer reveals distinct patterns across the X, Y, and Z axes that can be indicative of certain physical activities. The X-axis shows relatively low variability with a few spikes, suggesting limited lateral movement. The Y-axis displays periodic fluctuations with a significant peak, indicating a rhythmic up-and-down motion consistent with stepping actions. The Z-axis also shows a pattern of peaks and troughs, which aligns with vertical displacement typically associated with stepping. The presence of these regular oscillations across all axes suggests a repetitive motion that involves changes in elevation and balance, characteristic of a dynamic activity rather than a static one. The combination of these observations points towards an activity involving consistent and rhythmic movement, such as descending stairs. **Answer: descending stairs.**

step that leads to the wrong answer, or a claim the measurements contradict

supported claim or step that leads to the correct answer **Bold**: final answer (red wrong, green correct)

[…] output after the first answer, truncated. Claims were checked against the measurements above; steps against the gold answer.

Figure 10: **Qualitative example, PAMAP2**. OpenTSLM-SP misses the three shaded excursions and answers sitting. Per-channel normalization removes the gravity reference, so ascent and descent cannot be distinguished from the input; the example illustrates the missed motion rather than the direction.

Table 5: Composition of the synthetic training corpus. Variate and length ranges are observed after collection and filtering.

Generator family	Series	Share	Variates	Length
Chronos-2 multivariatizer	310,000	36.7%	3–10	512–8,192
KernelSynth (univariate)	299,999	35.5%	1	1,024–4,096
TimePFN LMC	215,000	25.5%	3–10	512–2,048
SymTime S2Generator	19,276	2.3%	3–10	1,024–4,096
Total	844,275	100%	1–10	512–8,192

`RationalQuadratic`, and `WhiteKernel` entries and one `ConstantKernel`. The sampled kernels are recursively combined using addition or multiplication with equal probability, and the series is drawn from the resulting GP prior on a fixed grid. Additive compositions superpose components, whereas multiplicative compositions allow one component to modulate another.

Linear Model of Coregionalization. Independent KernelSynth draws do not provide cross-channel dependence. The linear model of coregionalization (LMC) generator from TimePFN (Taga et al., 2025) instead expresses each channel as a mixture of shared latent processes. For a series with $D \sim \mathcal{U}\{3, \ldots, 10\}$ channels, we draw w from a Weibull distribution with shape 2 and set Q to $\mathrm{round}(10w + 1)$, clipped to $[\max(2, \lfloor D/20 \rfloor), D]$. Each of the Q latents receives an independently composed KernelSynth covariance and is sampled from the corresponding GP prior. The observed channels are convex mixtures $Z = WY$, with each row of $W \in \mathbb{R}^{D \times Q}$ drawn from $\mathrm{Dir}(\alpha \mathbf{1}_Q)$ and $\alpha \sim \mathcal{U}(0.1, 2.0)$. Small values of α produce sparse mixtures dominated by a few latents, whereas larger values distribute weight more evenly across the shared latents. Sampling α independently for each series yields a range of cross-channel dependence patterns.

Chronos-2 Multivariatizer. The largest family follows the synthetic-data design described in the Chronos-2 technical report (Ansari et al., 2025); because its implementation is not released, we reimplement the procedure from the report. Each latent univariate series is drawn uniformly from four generators: KernelSynth; a stationary AR(p) process with $p \sim \mathcal{U}\{1, \ldots, 5\}$, constructed by mapping partial autocorrelations in $(-0.95, 0.95)$ to AR coefficients through the Levinson recursion; a trend–seasonality–irregularity model combining a random polynomial trend, one to three sinusoids, and Gaussian noise; or an additive Holt–Winters state-space simulator with random smoothing constants and seasonal period. After standardising the latent series, we combine K of them using one of two multivariatizers, selected with equal probability. The *cotemporaneous* multivariatizer computes $Z = f(AY)$, where $A \in \mathbb{R}^{D \times K}$ has i.i.d. $\mathcal{N}(0, 1/K)$ entries. With probability 0.5, f is the identity; otherwise it is sampled uniformly from $\{\tanh, \sin, \mathrm{sgn}(x)\sqrt{|x|}\}$. This branch produces either linear or nonlinear instantaneous dependence within a series. The *sequential* multivariatizer instead computes $Z[d, t] = \sum_k a_{d,k} Y_k[t - \tau_{d,k}]$, with independently sampled lags $\tau_{d,k} \sim \mathcal{U}\{0, \ldots, 32\}$ and zero padding at the leading boundary, thereby introducing lead–lag relationships. Mixing shared non-stationary latents in the cotemporaneous branch can also produce cointegrated channels; we do not use a separate cointegration generator. Finally, with probability 0.3, between one and $\lfloor D/2 \rfloor$ channels are designated as known covariates, yielding a covariate-informed forecasting item.

SymTime. SymTime (Wang et al., 2025) provides a qualitatively different source of variation through its series–symbol generator, which samples a symbolic expression and then realises the corresponding series. Although the generator emits matched expression–series pairs, we retain only the series because Phase 1 applies next-patch rather than symbolic-recovery supervision.

Unification. The four generators emit different container formats. During collection, we cast each item to a `float32` array with shape `(variates, length)` and stream it to a unified Arrow shard. We reject items containing non-finite values; this filter removes 1,668 SymTime items and no items from the other families. The retained corpus occupies 30 GB in 16 Arrow files. At training time, all files share one corpus label so that the sampler treats them as a single mixture component. Per-item seeds make the generated corpora reproducible under the same software environment.

Table 6: **Phase 1 dataset statistics**, by generator family. Mean ± standard deviation over sampled series; real series are cut into 2,048-point windows. Spectral entropy is spectral flatness (0 a pure tone, 1 white noise); seasonal peak the largest |ACF| over lags $[2, L/4]$; trend R^2 the variance share of a least-squares line; cross-channel the mean |correlation| over variate pairs, undefined when univariate.

Source	n	Spectral entropy	ACF(1)	Seasonal peak	Trend R^2	Cross-channel
Chronos-2 multivariatizer	1,204	0.51±0.14	0.63±0.27	0.78±0.14	0.22±0.16	0.36±0.14
KernelSynth (univariate)	1,206	0.40±0.22	0.77±0.34	0.85±0.27	0.10±0.24	–
TimePFN LMC	1,202	0.45±0.14	0.66±0.28	0.80±0.16	0.07±0.12	0.59±0.20
SymTime S2Generator	1,204	0.75±0.16	0.37±0.22	0.41±0.21	0.10±0.13	0.24±0.19
Australian electricity (real)	1,200	0.40±0.02	0.98±0.00	0.93±0.02	0.03±0.02	0.80±0.07

Table 7: **Phase-2 corpora.** Natural split sizes and the balanced per-epoch count; multiplicity is how often an example is seen per epoch, with corpora larger than the target truncated to a fixed subset.

Corpus	Train	Val	Test	# Train Per Epoch	Multiplicity
HAR	68,542	8,718	8,222	40,000	0.58× (fixed subset)
Sleep-EDF	7,434	930	930	40,000	5–6×
ECG-QA	159,313	31,137	41,093	40,000	0.25× (fixed subset)
PAMAP2	4,207	234	234	40,000	9–10×
TimeMQA	33,764	1,876	1,876	40,000	1–2×
Total	273,260			200,000	

E.3 Dataset Statistics

Table 6 summarises the corpus with five scale-invariant descriptors, alongside the real forecasting corpus used in the same phase. Two things are visible. First, the families occupy different regions rather than being four parameterisations of one process: KernelSynth has the lowest spectral entropy, its composed kernels concentrating spectral power and leaving the series comparatively predictable, whereas SymTime has the highest together with by far the weakest periodicity, its symbolic realisations lying closer to noise. Second, the two multivariatizers impose different amounts of cross-channel dependence, the coregionalization mixture producing the more strongly coupled channels, which is what those mechanisms are for.

The real corpus falls inside the synthetic range on spectral entropy and trend, but outside it on both dependence measures: its half-hourly demand is more persistent than any synthetic family, and its five states are more tightly coupled than any generator produces. The synthetic corpus is, on the other hand, much more varied, its per-descriptor spread being an order of magnitude wider, since the real reference is a single system observed repeatedly whereas each synthetic draw resamples its own generating process. Pretraining on this mixture therefore buys breadth rather than fidelity to any one real corpus.

F Phase-2 Training Data

Phase 2 uses the five chain-of-thought corpora in Table 7, each rendered as question–rationale–answer triples: HAR (three-axis accelerometry), Sleep-EDF (single-channel EEG), ECG-QA (twelve-lead ECG), PAMAP2 (household and exercise activities from wearable accelerometry), and TimeMQA (open-ended questions over univariate series). TimeMQA is one training corpus but two evaluation columns, separated by its `question_format` field, which is why five corpora yield the six columns of Table 1.

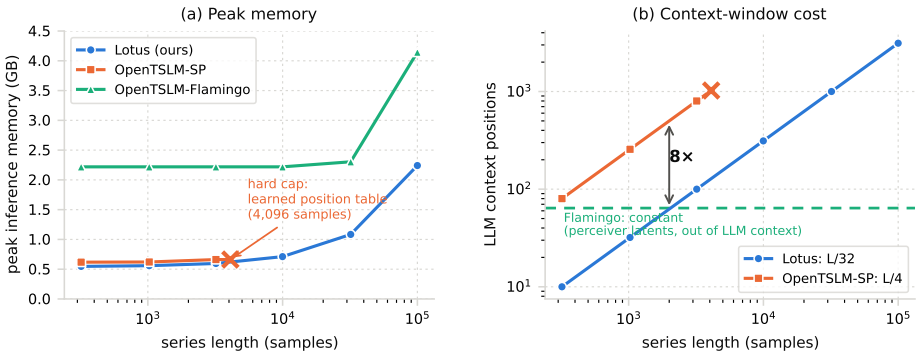


Figure 11: **Inference scaling with series length**, all baselines on Gemma3-270m (single series, one 10-token generation per point; peak allocated memory). Left: LOTUS is cheapest at every length; OpenTSLM-SP's line terminates at its architectural length ceiling. Right: context positions consumed, $L/32$ against $L/4$.

F.1 BALANCED MIXTURE.

Each corpus is resized to $40{,}000$ examples per epoch ($200{,}000$ in total): smaller corpora are repeated in full with a seeded shuffled remainder, larger ones are shuffled once and truncated to a subset that is reused in every epoch. Under the natural sizes, ECG-QA and HAR would supply 83% of the data and PAMAP2 1.5%; after balancing, each PAMAP2 example is seen nine to ten times per epoch while three quarters of ECG-QA is never used.

F.2 EVALUATION SUBSETS.

Sleep-EDF and PAMAP2 have validation and test splits of equal size (930 and 234). For ECG-QA we score the first $5{,}000$ test items, pinned by a manifest keyed on a hash of the gold answer so that every system is evaluated on the same subset.

G COMPUTATIONAL COST

How much of the backbone's context a series occupies decides both what it costs to read that series at inference and what it costs to train on it. Everything in this section is measured on one GPU and one backbone (Gemma3-270m) so that the baselines can be compared directly; the resources actually consumed by our training runs are reported in Appendix H.

G.1 READING A SERIES

Figure 11 measures what follows from this on one backbone. LOTUS is the cheapest interface at every length we tested and handles 100k samples in 2.2 GB. OpenTSLM-SP stops at $4{,}096$ samples, because its encoder uses a learned positional table of fixed length: longer inputs are not expensive, they are unrepresentable. OpenTSLM-Flamingo holds its token count nearly constant through a perceiver, but carries a fixed overhead of about 2.2 GB, four times ours at short lengths.

G.2 TRAINING ON A SERIES

Token count also sets the cost of a training step. Every number in Table 8 is measured at batch size 1, the Phase-2 batch, with each interface built at the geometry of the checkpoint we report and optimising its own trainable set; a step is one forward, backward and optimiser pass, averaged over 12 steps after warmup.

What Table 8 shows is a scaling property. Going from one channel of $1{,}500$ samples to twelve channels of $1{,}000$ multiplies our step time by 1.05 and our peak memory by 1.03. The same change costs OpenTSLM-SP $4.4\times$ in time and $6.9\times$ in memory, because a patch length of four turns twelve

Table 8: **Phase-2 cost per example on one GPU** (Gemma3-270m, batch 1). *tok* is the context the series occupies, constant for Flamingo, whose perceiver compresses to 64 latents; *s/ex* is one training step and *GB* peak training memory.

	Sleep-EDF (1×1500)			ECG-QA (12×1000)		
Interface	tok	s/ex	GB	tok	s/ex	GB
LOTUS (ours)	47	0.26	**0.63**	384	**0.28**	**0.65**
OpenTSLM-SP	375	**0.19**	2.44	3000	0.85	16.88
OpenTSLM-Flamingo	const.	0.22	9.14	const.	0.23	9.11
Tokenized	7239	3.48	22.78	7582	3.76	23.82

Table 9: **Phase 1 training runs**. Wall-clock time, allocation, and GPU-hours are from scheduler records. An epoch is one pass through the weighted sample drawn from the four corpora, not through every available example. Peak VRAM is the per-device maximum from a single-process V100 run at the listed per-device batch size.

Backbone	Encoder	GPUs	Batch		Peak	Wall clock	GPU-h	Epochs
			/device	eff.	VRAM			
Gemma3-270M	Transformer Decoder	2×4	16	128	31.3 GB	26 h	210	0.7
Qwen3-0.6B	Transformer Decoder	2×4	8	64	9.1 GB	5.0 d	960	2.1
Gemma3-1B	Transformer Decoder	4×4	8	128	15.2 GB	10.0 d	3,840	6.2
Total							5,010	

leads into 3,000 soft tokens where we use 384. On the twelve-lead column we therefore train $3.1\times$ faster than OpenTSLM-SP in $1/26$ of its memory, and $13.6\times$ faster than the serialized interface in $1/37$ of its memory. OpenTSLM-SP is faster on the short single-channel column (0.19 s against 0.26 s), and that ordering reverses as soon as the input is multichannel, which is the regime the interface exists for. OpenTSLM-Flamingo is input-invariant too, since its perceiver fixes the token count, but it pays $14\times$ our training memory and two orders of magnitude more trainable parameters. The serialized baseline looks invariant only because it throws data away: its harness decimates to 1,200 values across all channels, so the twelve-lead sample reaches the model with 90% of its samples gone and still costs 23.8 GB to train on.

Generation latency separates the baselines far less, and we report it so the efficiency claim is not over-read. At a 200-token budget every interface falls between 8.2 and 11.1 seconds per sample, and the spread does not change between the two input geometries: decoding is dominated by 200 sequential steps, so a 7,239-token prompt and a 47-token prompt finish within 15% of each other. The interface decides memory and reach at inference, not latency.

H COMPUTATIONAL RESOURCES

Table 9 reports what the Phase 1 predictive-grounding runs consumed.

H.1 HARDWARE AND PRECISION

All runs used NVIDIA V100 accelerators in four-GPU nodes, with data parallelism only, so each device held a complete model replica. V100s have no native bfloat16 arithmetic, so we used fp16 automatic mixed precision with gradient scaling.

H.2 TRAINING TIME

Each Phase 1 run is trained for at most ten days under the same early-stopping protocol. Validation loss is evaluated at regular intervals, and training stops when it does not improve for five consecutive validation checks. We retain the checkpoint with the lowest validation loss rather than the final checkpoint. Table 9 reports the actual wall-clock time and number of optimizer updates.

H.3 Vocabulary-dependent Memory

Peak VRAM is not monotonic in backbone size: Gemma3-270M uses 31.3 GB where Gemma3-1B uses 15.2 GB. The cross-entropy logit tensor explains almost all of it, since its shape is batch $\times$ length $\times$ $|\mathcal{V}|$. The two Gemma backbones share a 262,144-token vocabulary but run at per-device batches of 16 and 8, so the smaller one materialises twice the logits at a given sequence length, while Qwen3-0.6B pairs batch 8 with a smaller vocabulary of 151,936. Taking Qwen3-0.6B as the reference, batch $\times$ $|\mathcal{V}|$ predicts 1:1.73:3.45 for Qwen3-0.6B, Gemma3-1B, and Gemma3-270M against measured ratios of 1:1.67:3.44. Weights and activations contribute as well, but vocabulary size and per-device batch dominate. These single-process measurements omit the gradient buckets that data parallelism introduces, and the Gemma3-270M measurement enables the two-dimensional position-ID path that the original run predates, whose position tensor is negligible beside the logits.

H.4 Configurations Across Backbones

Gemma3-270M uses the time series encoder with LoRA adapters, training 3.17M parameters. Gemma3-1B and Qwen3-0.6B use the transformer encoder with LoRA adapters, training 14.73M and 12.65M, of which 6.52M and 5.05M are adapter weight, and halve the per-device batch to 8. Gemma3-1B uses twice the replicas of Qwen3-0.6B, which restores its effective batch to 128. These totals include the next-patch decoder, which Phase 2 discards; the Params column of Table 1 reports the Phase-2 trainable set and is correspondingly smaller.

I Prompt Templates

I.1 Phase 1

In phase 1, we use the caption on the channel block (Figure 12), giving the modality, the number of channels and time steps, and the mean and standard deviation of the raw series. Training uses the next-patch regression of Eq. 10, so no token in this prompt is a target. A forecasting batch therefore produces no cross-entropy term, which is why each Phase 1 batch is drawn from a single source.

Forecasting (electricity demand and the synthetic corpus)

PRE-PROMPT

(empty)

CHANNEL BLOCK

```
This is a univariate time series with 1 channel and 1024 time steps.
It has mean 6084.2410 and std 1174.1908.  <ts>...</ts>
```

POST-PROMPT

```
<eos>
```

Figure 12: Phase-1 forecasting prompt, shown for a 1,024-step window of electricity demand. A multivariate item from the synthetic corpus takes the same form with `multivariate`, its channel count, and one block per channel. The answer field is the end-of-sequence token; the training signal is the next-patch regression rather than any token in this prompt.

The two language corpora in Phase 1 follow the three-part schema of §I.2. Time-series QA places the item's question in the pre-prompt and `Predict the {task} Answer:` in the post-prompt, with the answer letter as the target. M4 captioning uses `You are an expert in time series analysis.` followed by `Please generate a detailed caption for this time-series, describing it as accurately as possible.`, with the reference caption as the target.

I.2 Phase 2

Every corpus uses the same three-part schema: a *pre-prompt* stating the task and instructions, one captioned block per channel, and a *post-prompt* fixing the answer format. In LOTUS and the soft-

prompt baselines each block holds the encoder's soft tokens, delimited by <ts>... </ts>; in the serialized baseline the same position holds the decimated values as text. Each caption names the channel and gives its mean and standard deviation, so normalisation statistics reach every interface as text. Figures 13–17 reproduce the templates verbatim; ellipses mark truncation only.

HAR

PRE-PROMPT

```
You are given accelerometer data in all three dimensions.  Your task
is to classify the activity based on analysis of the data.
Instructions:
- Begin by analyzing the time series without assuming a specific
label.
- Think step-by-step about what the observed patterns suggest
regarding movement intensity and behavior.
- Write your rationale as a single, natural paragraph -- do not use
bullet points, numbered steps, or section headings.
- Do **not** mention any class label until the final sentence.
Possible activity labels are:
biking, lying, running, sitting, standing, walking, walking_down,
walking_up.
- Make sure that your last word is the answer.  You MUST end your
response with "Answer:  "
```

CHANNEL BLOCKS (ONE PER AXIS)

```
The following is the accelerometer data on the x-axis, it has mean
-3.2434 and std 0.0474:  <ts>...</ts>
The following is the accelerometer data on the y-axis, it has mean
2.3132 and std 0.0550:  <ts>...</ts>
The following is the accelerometer data on the z-axis, it has mean
9.2017 and std 0.0639:  <ts>...</ts>
```

POST-PROMPT

```
Rationale:
```

Figure 13: Prompt template for HAR.

PAMAP2

PRE-PROMPT

```
You are given accelerometer data in all three dimensions.  Your task
is to classify the activity based on analysis of the data.
Instructions:
- Begin by analyzing the time series without assuming a specific
label.
- Think step-by-step about what the observed patterns suggest
regarding movement intensity and behavior.
- Write your rationale as a single, natural paragraph -- do not use
bullet points, numbered steps, or section headings.
- Do **not** mention any class label until the final sentence.
Possible activity labels are:
lying, sitting, standing, walking, running, cycling, nordic walking,
watching TV, computer work, car driving, ascending stairs, descending
stairs, vacuum cleaning, ironing, folding laundry, house cleaning,
playing soccer, rope jumping.
- Make sure that your last word is the answer.  You MUST end your
response with "Answer:  "
```

CHANNEL BLOCKS (ONE PER AXIS)

```
The following is the accelerometer data on the x-axis, it has mean
-3.2434 and std 0.0474:  <ts>...</ts>
The following is the accelerometer data on the y-axis, it has mean
2.3132 and std 0.0550:  <ts>...</ts>
The following is the accelerometer data on the z-axis, it has mean
9.2017 and std 0.0639:  <ts>...</ts>
```

POST-PROMPT

```
Rationale:
```

Figure 14: Prompt template for PAMAP2. The template matches HAR except for the eighteen-way label set.

Sleep-EDF

PRE-PROMPT

```
You are given a 30-second EEG time series segment.  Your task is to
classify the sleep stage based on analysis of the data.
Instructions:
- Analyze the data objectively without presuming a particular label.
- Reason carefully and methodically about what the signal patterns
suggest regarding sleep stage.
- Write your reasoning as a single, coherent paragraph.  Do not use
bullet points, lists, or section headers.
- Only reveal the correct class at the very end.
- Never state that you are uncertain or unable to classify the data.
You must always provide a rationale and a final answer.
```

CHANNEL BLOCK

```
This is the EEG signal, it has mean 0.0021 and std 0.9873:
<ts> ... </ts>
```

POST-PROMPT

```
Possible sleep stages are:
Wake, Non-REM stage 1, Non-REM stage 2, Non-REM stage 3, Non-REM stage
4, REM sleep
- Please now write your rationale.  Make sure that your last word is
the answer.  You MUST end your response with "Answer:  "
```

Figure 15: Prompt template for Sleep-EDF. Unlike the accelerometer corpora, the class list is placed in the post-prompt.

ECG-QA

PRE-PROMPT

```
You are an expert cardiologist analyzing an ECG (electrocardiogram).
Clinical Context:  82-year-old male patient.  12-lead ECG. clinical
recording.  recorded with AT-6 C 5.8.  Signal quality:  baseline drift
noted, static noise present, burst noise present, electrode artifacts
present.  extra beats detected during recording.  pacemaker present.
Your task is to examine the ECG signal and answer the following
medical question:
Question:  Does this ECG show burst noise?
Instructions:
- Begin by analyzing the time series without assuming a specific
answer.
- Think step-by-step about what the observed patterns suggest
regarding the cardiac condition.
- Write your rationale as a single, natural paragraph -- do not use
bullet points, numbered steps, or section headings.
- Do **not** mention any final answer until the very end.
- Consider the ECG morphology, intervals, and any abnormalities that
relate to the question.
```

CHANNEL BLOCKS (ONE PER LEAD, TWELVE IN TOTAL)

```
This is ECG Lead I, it has mean -0.0063 and std 0.1198:  <ts>...</ts>
... (Leads II, III, aVR, aVL, aVF, V1-V6)
```

POST-PROMPT

```
Based on your analysis of the ECG data, select your answer from the
following options:
no, yes
- Make sure that your last word is the answer.  You MUST end your
response with "Answer:  "
```

Figure 16: Prompt template for ECG-QA. The clinical-context paragraph is assembled from the record's metadata; the answer set in the post-prompt is that of the question template.

TimeMQA

PRE-PROMPT

```
You are given a time series from the Healthcare domain.  Answer the
following question.
Which data segment shows more volatility?  A) <the provided time
series> or B) [0.71, 0.72, 0.73, 0.73, 0.73, 0.73, 0.72, 0.72, 0.71,
0.7, 0.68, 0.67]
```

CHANNEL BLOCK

```
This is the time series, mean 0.5300 std 0.1352.  <ts>...</ts>
```

POST-PROMPT

```
Answer:
```

Figure 17: Prompt template for TimeMQA. The corpus is open-ended, so the post-prompt is a bare `Answer:`. In comparison questions the first segment enters through the time-series interface while the second is inlined as digits, so the model compares a soft-token series against a textual one.

I.3 TSRBench Evaluation

TSRBench is evaluated zero-shot, so the prompts are the benchmark's own rather than ours: no template is tuned for LOTUS and every system sees the same text. All fourteen subsets are multiple choice and share one answer instruction, which fixes the output format and separates the rationale from the graded answer; only the way the signal is introduced differs between the three templates below. Series enter at the <ts>... </ts> placeholders exactly as in Phase 2, so the interface is unchanged from training. Parsing takes the single letter inside `<answer>`; an item with no recoverable letter is scored incorrect.

Standard multiple choice (twelve of the fourteen subsets)

QUESTION AND CHANNEL BLOCKS

{question} Here are the time series '{name$_1$}': <ts>...</ts>. '{name$_2$}': <ts>...</ts>. ...

ANSWER INSTRUCTION (SHARED BY ALL SUBSETS)

```
Select from the options below:
A. {choice A}
B. {choice B}
...
Output your reasoning process in <think> tags and final answer as a
single letter in <answer> tags.  Format:
<think>Your reasoning here (less than 2048 tokens)</think>
<answer>A</answer>
```

Figure 18: Standard TSRBench template. Options are labelled A–G in the order the benchmark stores them.

Temporal relation reasoning

AS ABOVE, BUT WITH THE INDEX CHANNEL REMOVED

{question} Here are the time series '{name$_1$}': <ts>...</ts>. ...

Any channel named "Time Stamps" is dropped before the placeholders are emitted, so only measured channels are encoded; the answer instruction is unchanged.

Figure 19: Temporal relation reasoning. The subset ships an explicit timestamp channel, which would otherwise be encoded as though it were a signal.

Abductive reasoning

PRE-PROMPT

```
You are an expert in basketball game analysis.  Your task is to
perform abductive reasoning.
Given a sequence of past events, future events, and corresponding
time series data from a game, determine the most plausible event that
occurred in between to link them.
```

CONTEXT

```
-- CONTEXT --
Past Events (History):
- {time}:  {event} ...
...[A CRITICAL EVENT HAPPENED HERE] ...
Future Events:
- {time}:  {event} ...
Time Series Data (Win Probability):
Time | Team A Win Prob | Team B Win Prob
...
```

TASK, CHANNEL BLOCKS, THEN THE SHARED ANSWER INSTRUCTION

```
-- TASK --
```

```
{question} Here are the time series 'Team A Win Probability':
<ts>...</ts>.  'Team B Win Probability':  <ts>...</ts>.
```

Figure 20: Abductive reasoning. A window of ten steps either side of the critical moment is kept. The win-probability channels are given twice, once serialised in the table and once through the encoder, which is the benchmark's own design and the one setting where our interface receives the signal redundantly.