# TS-HAYSTACK: A MULTI-SCALE RETRIEVAL BENCHMARK FOR TIME SERIES LANGUAGE MODELS

**Nicolas Zumarraga**[*]
Agentic Systems Lab, ETH Zurich

**Thomas Kaar**
Agentic Systems Lab, ETH Zurich
Stanford Mussallem Center for Biodesign, Stanford University

**Ning Wang**
Agentic Systems Lab, ETH Zurich

**Maxwell A. Xu**
University of Illinois Urbana-Champaign
Google

**Max Rosenblattl**
Stanford Mussallem Center for Biodesign, Stanford University

**Markus Kreft**
Agentic Systems Lab, ETH Zurich

**Kevin O'Sullivan**
Agentic Systems Lab, ETH Zurich

**Paul Schmiedmayer**
Stanford Mussallem Center for Biodesign, Stanford University

**Patrick Langer**[†]
Agentic Systems Lab, ETH Zurich
Stanford Mussallem Center for Biodesign, Stanford University
Centre for Digital Health Interventions, ETH Zurich

**Robert Jakob**[†]
Agentic Systems Lab, ETH Zurich

## ABSTRACT

Time Series Language Models (TSLMs) are emerging as unified models for reasoning over continuous signals in natural language. However, long-context retrieval remains a major limitation: existing models are typically trained and evaluated on short sequences, while real-world time-series sensor streams can span millions of datapoints. This mismatch requires precise temporal localization under strict computational constraints, a regime that is not captured by current benchmarks. We introduce **TS-Haystack**, a long-context temporal retrieval benchmark comprising ten task types across four categories: direct retrieval, temporal reasoning, multi-step reasoning and contextual anomaly. The benchmark uses controlled needle insertion by embedding short activity bouts into longer longitudinal accelerometer recordings, enabling systematic evaluation across context lengths ranging from seconds to 2 hours per sample. We hypothesize that existing TSLM time series encoders overlook temporal granularity as context length increases, creating a task-dependent effect: compression aids classification but impairs retrieval of localized events. Across multiple model and encoding strategies, we observe a consistent divergence between classification and retrieval behavior. Learned latent compression preserves or improves classification accuracy at compression ratios up to 176×, but retrieval performance degrades with context length, incurring in the loss of temporally localized information. These results highlight the importance of architectural designs that decouple sequence length from computational complexity while preserving temporal fidelity.

## 1 INTRODUCTION

Edge devices generate vast continuous streams of time series data: a single 100Hz accelerometer produces over 8 million datapoints daily. However, extracting insights typically relies on statistical pipelines or specialized models (Bulling et al., 2014). Time Series Language Models (TSLMs) provide a path toward reasoning over sensor data by bridging the gap between high-dimensional

[*]Corresponding author: nzumarraga@ethz.ch
[†]Equal contribution as senior authors

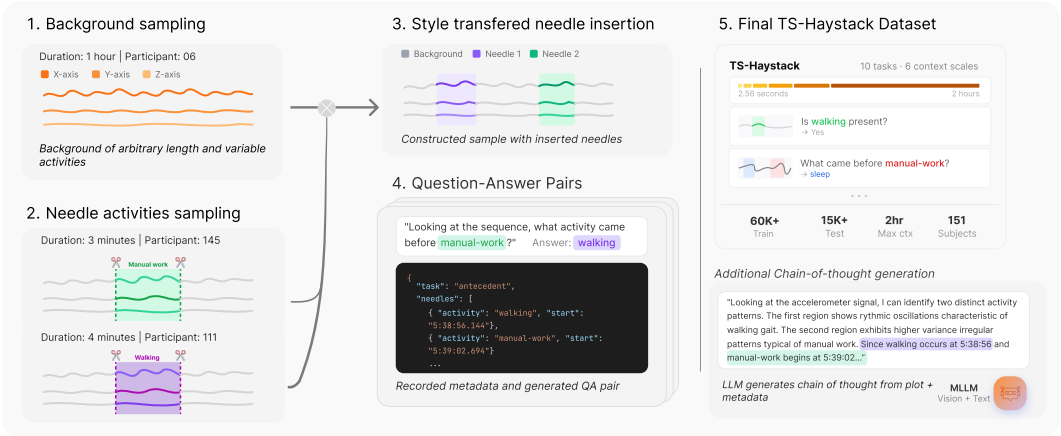



Figure 1: **TS-Haystack construction pipeline.** *Left:* A 1 hour window is sampled from a participant, activity needles ranging 1s – 6s are sampled at random. *Right:* Question-Answer pairs and chain-of-thought rationales are generated from plots and metadata-enriched prompt templates.

continuous signals and the embedding space of LLMs (Langer et al., 2025; Jin et al., 2024; Wang et al., 2025) using specialized time series encoders to bridge the dimensionality gap. However, these models are rarely trained on sample windows exceeding 1,000 datapoints, orders of magnitude below real-world requirements. We hypothesize that existing TSLM encoders do not retain sufficient temporal granularity at longer contexts lengths, and their compression mechanisms discard information needed for retrieval tasks.

Existing TSLM architectures face a fundamental trade-off between precise temporal representations and scalability. While full self-attention encoders attend to the whole sequence's information, it incurs in a quadratic computational cost (Nie et al., 2023; Vaswani et al., 2017). In contrast, fixed latent-space architectures improve efficiency through compression, but at the expense of temporal resolution. We also hypothesize a task-dependent effect. In *classification*, latent resampling may benefit by filtering noise; for *retrieval* tasks requiring localization of specific events, the same compression may marginalize away distinguishing patterns. Although benchmarks probing long-context retrieval exist for LLMs (Kuratov et al., 2024; Kamradt, 2023; Wang et al., 2024) and recent efforts have enabled evaluating time series foundation models in classification and forecasting tasks (Goswami et al., 2024) and statistical understanding (Cai et al., 2024), to our best knowledge, no benchmark currently enables evaluation of retrieval performance over long-contexts for TSLMs. To fill this gap, we introduce **TS-Haystack**, a retrieval benchmark adapting the *needle-in-a-haystack* paradigm to time series using real activity samples: short activity bouts (*needles*) are inserted into real accelerometer recordings spanning up to 2 hours (*haystacks*), where models must retrieve and reason about them. The benchmark comprises ten tasks with distinct requirements, enabling fine-grained diagnosis of where and why temporal understanding breaks down as context grows. We also evaluate different TSLM architectures on the Capture24 (Chan et al., 2024) classification dataset at context lengths ranging 2.56 seconds to 15 minutes. Further ablation introducing an *oracle* model variant allows us to investigate the origin of our long-context retrieval benchmark performance.

**Contributions.** We advance long-context TSLM evaluation with three contributions: **(1)** We evaluate TSLMs on long-context tasks, finding that latent resampling architectures maintain or even improve classification accuracy at compression ratios exceeding 176× but retrieval performance degrades over the same range. **(2)** We introduce **TS-Haystack**, a retrieval benchmark containing ten task types across four categories, constructed via controlled needle insertion into Capture24 recordings (Chan et al., 2024), with chain-of-thought reasoning variants to support TSLM training and evaluation at scale. **(3)** We propose a needle insertion protocol using real data insertion in long context backgrounds, enabling generation of retrieval benchmarks from annotated time series datasets.

## 2 Background

We categorize TSLM architectures based on how they encode time series into the language model's input space: *full-resolution adapters* that apply self-attention over all whole sequence length, and *latent resampling* methods, which compress them into a fixed number of tokens.

**Full-Resolution Adapters.** Approaches such as Softprompting or Time-LLM (Jin et al., 2024), map the sequence of $N$ patch embeddings to the language space. However, this fidelity comes at a prohibitive computational cost ($\mathcal{O}(N^2)$) either through the LLM's extended context window or within the time series encoder, which must perform self-attention over the full patch sequence

before projection (Wang et al., 2025). Consequently, these methods struggle to scale to larger context lengths due to computational cost and large memory requirements.

**Latent Resampling.** TSLM architectures leveraging cross-attention mechanisms (Vaswani et al., 2017), such as OpenTSLM-Flamingo (Langer et al., 2025), interpose a *Perceiver Resampler* (Jaegle et al., 2021) or similar components between the patch encoder and the LLM. Formally, a time series input of length $L$ is divided into $N = L/P$ non-overlapping patches. A fixed set of $M$ learned latent queries $\mathbf{Q} \in \mathbb{R}^{M \times d}$ cross-attend to these $N$ patch embeddings, producing an output $\mathbf{Z}'$ of constant size $M$ independent of $N$, enabling linear scaling with respect to input length. However, we hypothesize an information bottleneck: as input length grows, the compression ratio $N/M$ increases, forcing each token to summarize more information without prior knowledge of the specific downstream question. In OpenTSLM-Flamingo, $M$ defaults to 64 latent queries, making the $N/M$ compression ratio close to $176\times$ for a 15 minute recording sampled at 50Hz and patch size 4.

## 3 METHODS

In this section, we describe how we created the classification and TS-Haystack benchmark from the Capture24 dataset, including task taxonomy, dataset construction pipeline, needle insertion protocol, and training setup for the two evaluated architectures.

Table 1: Overview of TS-Haystack tasks grouped by category, with increasing difficulty.

| Task | Description | Task Requirement | Answer |
|---|---|---|---|
| ***Direct Retrieval*** | | | |
| Existence | Detect presence of a specific activity | Pattern recognition | Boolean |
| Localization | Temporally ground an activity | Recognition, Localization | Time range |
| Counting | Enumerate activity occurrences | Recognition, Enumeration | Integer |
| ***Temporal Reasoning*** | | | |
| Ordering | Compare activity temporal order | Localization, comparison | Boolean |
| State Query | Identify regime around local event | Cross-scale integration | Category |
| Antecedent | Identify activity preceding a target | Localization, Adjacency | Category |
| ***Multi-Step Reasoning*** | | | |
| Comparison | Compare durations across gaps | Localization, comparison | Time range |
| Multi-Hop | Locate $K$-th target relative to anchor | Localization, counting | Time range |
| ***Contextual Anomaly*** | | | |
| Anomaly Det. | Detect cross-regime activity | Contextual reasoning | Bool. + cat. |
| Anomaly Loc. | Detect and localize anomaly | Contextual, Localization | Bool. + cat. + time |

Note: Det. - Detection; Loc. - Localization

**Classification on Capture24.** We first evaluate both architectures on activity classification using Capture24 (Chan et al., 2024), a dataset of 151 participants wearing triaxial wrist accelerometers at 100 Hz for approximately 24 hours. We use the Willetts et al. (2018) annotation scheme as the class mapping for the annotated Capture24 dataset, providing 10 activity classes. Full details on class distributions, sampling budgets, and training protocol are provided in Appendix A.

**TS-Haystack Benchmark.** Following the *needle-in-a-haystack* paradigm (Kamradt, 2023), we insert short activity bouts (*needles*) into longer background recordings (*haystacks*) and task the model with retrieving and reasoning about them. The benchmark comprises ten tasks of increasing complexity (Table 1), generated at six context lengths . For each combination of context length and task type, we generate 1,000 training, 150 validation, and 150 test samples from 20 manually crafted question templates per task. Full task specifications and examples are provided in Appendices B.6 and B.

**Needle Insertion.** Needles are randomly sampled and inserted using per-channel mean alignment to adjust for participant-specific sensor biases while preserving activity dynamics, with cosine blending at insertion boundaries for smooth transitions (Appendix B.2). We validate that this protocol produces statistically undetectable insertions via classifier-based testing (Appendix B.3).

**Training Details.** We select recent open-source implementations of the two architectural categories: OpenTSLM-Flamingo (Langer et al., 2025) and ITFormer (Wang et al., 2025), using Llama-3.2-1B (Grattafiori et al., 2024) as the common LLM backbone. Models are trained jointly on all tasks and context lengths, using chain-of-thought rationales generated with *gpt-4.1-mini-2025-04-14* (OpenAI, 2023). ITFormer encountered out-of-memory errors at context lengths $\geq$40,000 datapoints; these were excluded from training and evaluation (Figure 2).

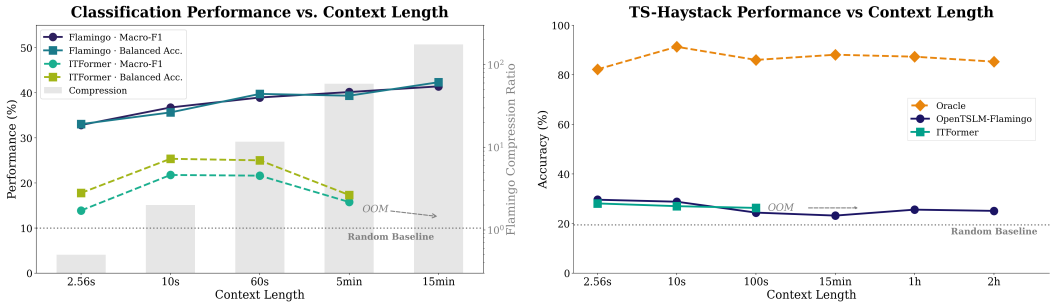
**Oracle Variant.** Since both the time series encoder and the language model contribute to the final answer, poor retrieval performance could stem from either component. We train an *Oracle Variant* of OpenTSLM-Flamingo receiving ground-truth activity segmentation as text metadata alongside the time series input, bypassing the need for the encoder to provide grounded activity recognition in its output representations. If the oracle maintains high retrieval accuracy across context lengths, the bottleneck lies in the time series encoding rather than in reasoning difficulty (see Appendix B.5).

## 4 RESULTS

In the following, we assess scaling behavior on classification and TS-Haystack retrieval tasks.

**Scaling Behavior.** Figure 2 displays classification and retrieval performance across context lengths. On Capture24 classification (Figure 2(a)), OpenTSLM-Flamingo's performance improves consistently despite compression ratios growing from 0.5× to 176×: Macro-F1 increases from 32.9% to 41.4%, a relative gain of 25.8%. Despite preserving full temporal resolution, ITFormer fails to match this trend: Macro-F1 peaks at 21.8% (10s) and degrades back to 15.8% at 5 minutes context length before running out of memory at 15 minutes. Further details are provided in Appendix A. On retrieval tasks (Figure 2(b)), OpenTSLM-Flamingo starts at 29.6% (2.56 s) but drops sharply to 23.2% at 15 minutes, after which performance plateaus near 25%. Despite maintaining full temporal resolution, ITFormer averages 27.2% accuracy across its three feasible context lengths, performing similarly to OpenTSLM-Flamingo in its feasible context lengths.

**Oracle Upper Bound.** Our oracle variant achieves 86.8% overall accuracy and, critically, maintains stable performance across context lengths (82.2%–91.3%), indicating that the LLM backbone possesses sufficient reasoning capacity.




(a) Classification (Capture24) (b) Retrieval (TS-Haystack)

Figure 2: **Scaling behaviors.** (a) Capture24 classification: Macro-F1 and Balanced Accuracy increase with context length. (b) TS-Haystack: Retrieval accuracy degrades with context length.

Table 2: TS-Haystack accuracy by task and context length. ITFormer encounters OOM errors beyond 100s; long-context results are OpenTSLM-Flamingo only. Best results per time series length in **bold**.

| | | 2.56s | | 10s | | 100s | | 15min | 1h | 2h |
|---|---|---|---|---|---|---|---|---|---|---|
| **Task** | Rand % | Flam. | IT | Flam. | IT | Flam. | IT | Flam. | Flam. | Flam. |
| *Direct Retrieval* | | | | | | | | | | |
| Existence | *50.0* | **57.3** | 54.7 | 47.3 | **52.7** | **54.0** | 53.3 | 53.3 | 51.3 | 50.0 |
| Localization | *~1.5* | 2.7 | **5.3** | **2.0** | 1.3 | 6.0 | **7.3** | 4.0 | 3.3 | 2.7 |
| Counting | *20.0* | 19.3 | **28.7** | 17.3 | **22.0** | 20.7 | **22.7** | 16.0 | 20.7 | 18.7 |
| *Temporal Reasoning* | | | | | | | | | | |
| Ordering | *50.0* | **46.7** | 45.3 | **52.7** | 42.7 | 38.7 | **46.7** | 41.3 | 54.0 | 50.7 |
| State Query | *10.0* | **65.3** | 40.7 | **68.0** | 52.0 | **24.7** | **24.7** | 22.0 | 31.3 | 34.7 |
| Antecedent | *10.0* | **7.3** | 6.0 | **13.3** | 8.0 | 11.3 | **13.3** | 8.7 | 9.3 | 9.3 |
| *Multi-Step Reasoning* | | | | | | | | | | |
| Comparison | *~1.5* | **8.7** | 8.0 | **9.3** | 6.0 | 6.7 | **8.7** | 6.0 | 8.0 | 3.3 |
| Multi-Hop | *~1.5* | 2.7 | **4.7** | 0.7 | **6.7** | 2.7 | **6.0** | 3.3 | 1.3 | 2.7 |
| *Contextual Anomaly* | | | | | | | | | | |
| Anomaly Det. | *50.0* | 52.0 | **56.0** | 46.7 | **54.0** | 45.3 | **51.3** | 50.0 | 48.0 | 54.7 |
| Anomaly Loc. | *~0.75* | **34.0** | 32.0 | **30.7** | 24.7 | **34.0** | 29.3 | 27.3 | 28.7 | 24.0 |
| **Average** | *~19.5* | **29.6** | 28.1 | **28.8** | 27.0 | 24.4 | **26.3** | 23.2 | 25.6 | 25.1 |

**Discussion.** With our work, we seek to improve understanding of TSLM architectures with respect to task type and context length, expanding existing benchmarks from annotated short intervals (Goswami et al., 2024) and synthetic long contexts (Cai et al., 2024) to labeled long-context retrieval.

Our results confirm that TSLMs experience task-dependent performance: OpenTSLM-Flamingo improves in classification as context length grows, yet retrieval performance degrades over the same range. Contrary to our initial hypothesis, retrieval performance does not linearly degrade with context length; performance drops from 29.6% (2.56 s) to 23.2% (15 min) then plateaus near 25% through increasing context lengths up to 2 hours. This non-monotonic degradation suggests that the bottleneck is not purely one of information loss through compression. State Query performance drops from 65.3% at 2.56 s to 24.7% at 15 min, contributing to 78% of the OpenTSLM-Flamingo degradation over the same period. At short lengths, State Query resembles a classification problem: the background is dominated by one activity regime that frequently coincides with the needle's context, making correct answers achievable through background classification. Other tasks requiring localization or contextualized short-interval identification remain difficult across all context lengths, suggesting a shared bottleneck even at short sequences. Our oracle experiment suggests that this degradation originates within the time series encoding and projection layers: when ground-truth segmentation is provided as text, retrieval accuracy remains stable across all context lengths (82.2%–91.3%), indicating sufficient reasoning capacity in the LLM backbone. We note that ITFormer, despite using full-attention, does not over-perform even at short contexts. An extended discussion is provided in Appendix B.4 and results should be interpreted with caution. However, the consistently flat accuracy suggests that preserving full-attention in the encoder is insufficient for retrieval. All findings point to a key design principle: temporal granularity must be decoupled from computational complexity *throughout* the architecture, not only at the adapter interface.

In addition, our work was validation on a single time series domain. Future work extending to additional sensor modalities (e.g., ECG, IMU, environmental monitoring) and applications remains crucial to establish our framework as a generalizable approach for time series language retrieval benchmark generation. We introduce TS-Haystack as a diagnostic benchmark for future research in long-context architectures (Gu & Dao, 2023; Behrouz et al., 2025), incorporating task and context-length adaptation.

## REFERENCES


Ali Behrouz, Peilin Zhong, and Vahab Mirrokni. Titans: Learning to memorize at test time. In *Advances in Neural Information Processing Systems*, 2025.

Andreas Bulling, Ulf Blanke, and Bernt Schiele. A tutorial on human activity recognition using body-worn inertial sensors. *ACM Computing Surveys*, 46(3):1–33, 2014. doi: 10.1145/2499621.

Yifu Cai, Arjun Choudhry, Xing Hu, and Artur Dubrawski. Timeseriesexam: A time series understanding exam. *arXiv preprint arXiv:2410.14752*, 2024.

Shing Chan, Hang Yuan, Catherine Tong, Aidan Acquah, Abram Schonfeldt, Jonathan Gershuny, and Aiden Doherty. CAPTURE-24: A large dataset of wrist-worn activity tracker data collected in the wild for human activity recognition. *Scientific Data*, 11(1):1135, 2024.

Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 785–794, 2016.

Mononito Goswami, Konrad Szafer, Arjun Choudhry, Yifu Cai, Shuo Li, and Artur Dubrawski. MOMENT: A family of open time-series foundation models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 16115–16152. PMLR, 2024.

Aaron Grattafiori et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

Andrew Jaegle, Felix Gimeno, Andrew Brock, Andrew Zisserman, Oriol Vinyals, and João Carreira. Perceiver: General perception with iterative attention. In *International Conference on Machine Learning*, pp. 4651–4664. PMLR, 2021.

Ming Jin, Shiyu Wang, Linton Ma, Zhiyu 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. URL https://arxiv.org/abs/2310.01728.

Greg Kamradt. Needle in a haystack - pressure testing LLMs. https://github.com/gkamradt/LLMTest_NeedleInAHaystack, 2023. Accessed: 2025.

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, Vienna, Austria, 2025. Association for Computational Linguistics.

Yuri Kuratov, Aydar Bulatov, Petr Anokhin, Ivan Rodkin, Dmitry Sorokin, Artyom Sorokin, and Mikhail Burtsev. BABILong: Testing the limits of LLMs with long context reasoning-in-a-haystack. In *Advances in Neural Information Processing Systems*, volume 37, pp. 106519–106554, 2024.

Patrick Langer et al. OpenTSLM: Time-series language models for reasoning over multivariate medical text- and time-series data. *arXiv preprint arXiv:2510.02410*, 2025.

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*, 2023.

OpenAI. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Hengyi Wang, Haizhou Shi, Shiwei Tan, Weiyi Qin, Wenyuan Wang, Tunyu Zhang, Akshay Namber, Tanuja Yu, and Hao Liu. Multimodal needle in a haystack: Benchmarking long-context capability of multimodal large language models. *arXiv preprint arXiv:2406.11230*, 2024.

Yilin Wang, Peixuan Lei, Jie Song, Yuzhe Hao, Tao Chen, Yuxuan Zhang, Lei Jia, Yuanxiang Li, and Zhongyu Wei. Itformer: Bridging time series and natural language for multi-modal qa with large-scale multitask dataset. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, 2025. URL https://arxiv.org/abs/2506.20093.

Matthew Willetts, Sven Hollowell, Louis Aslett, Chris Holmes, and Aiden Doherty. Statistical machine learning of sleep and physical activity phenotypes from sensor data in 96,220 UK Biobank participants. *Scientific Reports*, 8(1):7961, 2018. doi: 10.1038/s41598-018-26174-1.

# A Classification Experiment Details

## A.1 Experimental Setup

**Model architecture.** We use the Flamingo variant of OpenTSLM (Langer et al., 2025) with Llama-3.2-1B as the backbone LLM. To accommodate variable context lengths beyond the original training configuration, we extend the positional encoding of the Perceiver Resampler via linear interpolation. Critically, these interpolated positional embeddings are *unfrozen* during fine-tuning, allowing the model to adapt to longer temporal contexts.

**Dataset alignment.** We downsample Capture24 from 100Hz to 50Hz to match the sampling rate of the HAT accelerometer dataset used for OpenTSLM pretraining. The minimum context length of 2.56 seconds (128 timesteps) corresponds to the original OpenTSLM training window, providing a consistent baseline for measuring scaling behavior. This alignment ensures that performance differences across context lengths reflect genuine scaling effects rather than distribution shift from mismatched preprocessing.

**Label assignment.** Since longer context windows may span multiple activities, we determine classification labels via majority voting over the ground-truth annotation timeline. We require at least 60% temporal coverage by a single activity class for a sample to receive that label; windows without a clear majority are excluded from evaluation.

**Training protocol.** To ensure no data leakage, we first split participants (total of 151) into train (100), validation (25) and test (26) splits at random. We adopt a curriculum learning approach, training sequentially from shorter to longer contexts. For each context length, we use a maximum sample budget: 80,000 training, 15,000 validation, and 15,000 test samples (see Tables 3 and 4 for the relationship between available and sampled data). Models are trained for up to 50 epochs with early stopping based on validation loss, using a learning rate of $5 \times 10^{-5}$ with AdamW optimizer.

**Dataset size.** While longer context lengths up to 1 hour were initially included in training and yielded increasingly higher Macro-F1 scores, evaluation at these scales is limited by the scarcity of sufficiently long single-activity bouts in the source data; we therefore exclude these lengths from our main results to ensure statistically robust comparisons. At 5 and 15 minutes context length, the classification evaluation relies on smaller test sets (4,697 and 1,473 samples) due to the scarcity of contiguous single-activity segments. Performance at these context lengths should be interpreted with caution; yet the consistent upward trend suggests a robust positive effect.

## A.2 Dataset Statistics

Table 3 reports the total number of qualifying activity bouts (i.e. windows meeting the 60% majority criterion) available in the Capture24 dataset for each context length and split. As context length increases, the number of non-overlapping windows that satisfy the majority-vote criterion drops sharply: the 2.56 s setting yields over 3.5 M bouts, whereas the 15 min setting provides fewer than 6,500.

To maintain a consistent training signal across context lengths despite the large variation in available data, we cap sampling at 80,000 training, 15,000 validation, and 15,000 test samples per context length (Table 4). When the pool exceeds the cap, samples are drawn uniformly at random (seed 42); otherwise the full set is used. This strategy ensures that shorter-context models are not disproportionately advantaged by larger datasets, while longer-context settings retain every available sample.

Tables 5– 7 report the per-class label distributions for the sampled training, validation, and test sets, respectively. Across all splits and context lengths, *sleep* and *sitting* together account for roughly 70%

Table 3: Total number of qualifying activity bouts in Capture24 per context length and data split, before any sampling.

| Context | Train | Val | Test | Total |
|---|---|---|---|---|
| 2.56 s | 2,488,532 | 542,673 | 566,768 | 3,597,973 |
| 10 s | 636,107 | 138,644 | 144,841 | 919,592 |
| 60 s | 106,515 | 23,292 | 24,295 | 154,102 |
| 5 min | 20,724 | 4,502 | 4,697 | 29,923 |
| 15 min | 6,489 | 1,406 | 1,473 | 9,368 |
| **Avg. split** | **69.2%** | **15.1%** | **15.7%** | **100%** |

Table 4: Actually sampled dataset sizes per context length and split. "Random" indicates uniform subsampling to the budget cap; "full" indicates all available bouts were used.

| | Train | | Val | | Test | |
|---|---|---|---|---|---|---|
| **Context** | $N$ | **Method** | $N$ | **Method** | $N$ | **Method** |
| 2.56 s | 80,000 | random | 15,000 | random | 15,000 | random |
| 10 s | 80,000 | random | 15,000 | random | 15,000 | random |
| 60 s | 80,000 | random | 15,000 | random | 15,000 | random |
| 5 min | 20,724 | full | 4,502 | full | 4,697 | full |
| 15 min | 6,489 | full | 1,406 | full | 1,473 | full |

of samples, whereas minority classes such as *bicycling*, *sports*, and *manual-work* each contribute fewer than 2%. This imbalance is further exacerbated at longer contexts, where rare activities are increasingly unlikely to sustain the 60% majority threshold over extended windows.

Table 5: Training set class distribution (number of samples) across context lengths on Capture24. Percentages are computed within each context length.

| Activity | 2.56 s | 10 s | 60 s | 5 min | 15 min |
|---|---|---|---|---|---|
| sleep | 29,538 (36.9%) | 30,048 (37.6%) | 29,481 (36.9%) | 7,851 (37.9%) | 2,601 (40.1%) |
| sitting | 28,712 (35.9%) | 28,450 (35.6%) | 28,661 (35.8%) | 7,637 (36.9%) | 2,515 (38.8%) |
| household-chores | 5,616 (7.0%) | 5,460 (6.8%) | 5,540 (6.9%) | 1,383 (6.7%) | 381 (5.9%) |
| walking | 5,107 (6.4%) | 4,944 (6.2%) | 5,114 (6.4%) | 1,080 (5.2%) | 220 (3.4%) |
| mixed-activity | 3,324 (4.2%) | 3,422 (4.3%) | 3,440 (4.3%) | 869 (4.2%) | 263 (4.1%) |
| vehicle | 2,849 (3.6%) | 2,883 (3.6%) | 2,882 (3.6%) | 747 (3.6%) | 231 (3.6%) |
| standing | 2,515 (3.1%) | 2,485 (3.1%) | 2,531 (3.2%) | 559 (2.7%) | 94 (1.4%) |
| manual-work | 1,074 (1.3%) | 1,063 (1.3%) | 1,091 (1.4%) | 281 (1.4%) | 89 (1.4%) |
| bicycling | 848 (1.1%) | 839 (1.0%) | 848 (1.1%) | 209 (1.0%) | 59 (0.9%) |
| sports | 417 (0.5%) | 406 (0.5%) | 412 (0.5%) | 108 (0.5%) | 36 (0.6%) |
| **Total** | **80,000** | **80,000** | **80,000** | **20,724** | **6,489** |

### A.3 Per-Class Performance

**Class imbalance.** The Capture24 test set exhibits severe class imbalance: *sleep* and *sitting* together account for over 70% of all windows at every context length, while minority classes such as *bicycling*, *manual-work*, and *standing* each represent fewer than 3% of samples (Table 5). To mitigate that performance on rare activities is not masked by majority-class accuracy while preserving a significant number of samples for training and evaluation at longer context lengths, we report **Macro-F1** and **Balanced Accuracy** as our primary evaluation metrics Bulling et al. (2014). Macro-F1 computes the unweighted mean of per-class F1 scores, giving equal importance to every activity regardless of prevalence. Balanced Accuracy averages per-class recall, providing a complementary view that is similarly insensitive to class frequency. These metrics penalize models that achieve high overall accuracy by defaulting to majority-class predictions, a pattern evident in the prediction distributions of Table 7, where the model over-predicts *sleep* and *sitting* across all context lengths.
Table 8 reports per-class F1 scores alongside Macro-F1 and Balanced Accuracy across context lengths.

### A.4 Training Statistics

Table 9 compares class-imbalance-aware metrics across context lengths for OpenTSLM-Flamingo and ITFormer.

Table 6: Validation set class distribution (number of samples) across context lengths on Capture24. Percentages are computed within each context length.

| Activity | 2.56 s | 10 s | 60 s | 5 min | 15 min |
|---|---|---|---|---|---|
| sleep | 5,545 (37.0%) | 5,553 (37.0%) | 5,502 (36.7%) | 1,715 (38.1%) | 569 (40.5%) |
| sitting | 5,521 (36.8%) | 5,462 (36.4%) | 5,513 (36.8%) | 1,695 (37.6%) | 554 (39.4%) |
| walking | 1,032 (6.9%) | 1,047 (7.0%) | 1,040 (6.9%) | 268 (6.0%) | 52 (3.7%) |
| household-chores | 794 (5.3%) | 761 (5.1%) | 816 (5.4%) | 222 (4.9%) | 56 (4.0%) |
| mixed-activity | 634 (4.2%) | 634 (4.2%) | 647 (4.3%) | 185 (4.1%) | 59 (4.2%) |
| standing | 615 (4.1%) | 632 (4.2%) | 613 (4.1%) | 153 (3.4%) | 31 (2.2%) |
| vehicle | 562 (3.7%) | 633 (4.2%) | 578 (3.9%) | 176 (3.9%) | 57 (4.1%) |
| bicycling | 216 (1.4%) | 197 (1.3%) | 217 (1.4%) | 67 (1.5%) | 21 (1.5%) |
| sports | 50 (0.3%) | 49 (0.3%) | 37 (0.2%) | 13 (0.3%) | 4 (0.3%) |
| manual-work | 31 (0.2%) | 32 (0.2%) | 37 (0.2%) | 8 (0.2%) | 3 (0.2%) |
| **Total** | **15,000** | **15,000** | **15,000** | **4,502** | **1,406** |

Table 7: Test set class distribution (number of samples) across context lengths on Capture24. Percentages are computed within each context length. Longer contexts yield fewer non-overlapping windows from the fixed-length recordings, reducing total test samples at 5 min and 15 min.

| Activity | 2.56 s | 10 s | 60 s | 5 min | 15 min |
|---|---|---|---|---|---|
| sleep | 5,424 (36.2%) | 5,522 (36.8%) | 5,423 (36.2%) | 1,758 (37.4%) | 584 (39.6%) |
| sitting | 5,183 (34.6%) | 5,087 (33.9%) | 5,186 (34.6%) | 1,659 (35.3%) | 555 (37.7%) |
| household-chores | 1,379 (9.2%) | 1,377 (9.2%) | 1,404 (9.4%) | 418 (8.9%) | 117 (7.9%) |
| walking | 1,041 (6.9%) | 1,065 (7.1%) | 1,061 (7.1%) | 272 (5.8%) | 48 (3.3%) |
| vehicle | 659 (4.4%) | 669 (4.5%) | 621 (4.1%) | 212 (4.5%) | 64 (4.3%) |
| mixed-activity | 566 (3.8%) | 559 (3.7%) | 580 (3.9%) | 174 (3.7%) | 55 (3.7%) |
| standing | 407 (2.7%) | 397 (2.6%) | 419 (2.8%) | 101 (2.2%) | 22 (1.5%) |
| sports | 171 (1.1%) | 169 (1.1%) | 164 (1.1%) | 54 (1.1%) | 16 (1.1%) |
| manual-work | 139 (0.9%) | 117 (0.8%) | 116 (0.8%) | 40 (0.9%) | 11 (0.7%) |
| bicycling | 31 (0.2%) | 38 (0.3%) | 26 (0.2%) | 9 (0.2%) | 1 (0.1%) |
| **Total** | **15,000** | **15,000** | **15,000** | **4,697** | **1,473** |

**OpenTSLM-Flamingo compression ratio.** In OpenTSLM-Flamingo, a *CNN-Tokenizer* divides each input channel into non-overlapping patches of size $p = 4$ samples, yielding $N = L/p$ patches for a sequence of $L$ timesteps. The Perceiver Resampler (Langer et al., 2025) then compresses all $N$ patch embeddings into a fixed set of $M = 64$ latent tokens via cross-attention, resulting in a compression ratio of $N/M$. At short contexts (2.56s, $N = 32$), the number of patches is less than the number of latent tokens, so no compression occurs. As context length increases, compression grows linearly: at 15 minutes (45,000 timesteps), 11,250 patches are compressed into 64 tokens, a 176$\times$ reduction in sequence length presented to the LLM. Despite this compression, classification performance improves with context length (Figure 2(a)), suggesting the resampler effectively distills activity-relevant features while filtering per-timestep noise.

**ITFormer baseline.** We additionally evaluate ITFormer (Wang et al., 2025) as a full-resolution baseline. We adopt the original architecture with a 4-layer time series encoder ($d_{\text{model}}$=512, 8 heads) and a 2-layer cross-modal prefix module ($d_{\text{model}}$=896, 16 heads, 25 prefix tokens). Patches use a non-overlapping window of 60 samples at 50 Hz (1.2 s per patch). We train with a learning rate of $1 \times 10^{-5}$, batch size 8, weight decay $1 \times 10^{-6}$, and FP16 mixed precision for 3 epochs. The LLM backbone is frozen and only the time series encoder and prefix module are trained. At 15 minutes (900 s context), training encountered out-of-memory errors on an NVIDIA RTX PRO 6000 (96 GB).

# B TS-Haystack details

## B.1 Related Benchmark Literature

TS-Haystack builds upon and distinguishes itself from prior work in time series question answering and long-context evaluation. We position our benchmark relative to existing approaches across three dimensions: task design philosophy, data construction methodology, and evaluation scope.

**Time Series Foundation Models.** MOMENT (Goswami et al., 2024) releases the first open-source foundation models for general-purpose time series analysis, pre-trained on the Time Series Pile, a large multi-domain collection of public time series. Their benchmark evaluates five tasks: forecasting, classification, anomaly detection, and imputation, with emphasis on limited supervision settings.

Table 8: OpenTSLM-Flamingo per-class F1 scores (%) across context lengths on Capture24.

| **Activity** | **2.56 s** | **10 s** | **60 s** | **5 min** | **15 min** |
|---|---|---|---|---|---|
| sleep | 83.2 | 86.2 | 85.1 | 86.8 | 88.0 |
| sitting | 60.4 | 67.0 | 68.3 | 73.3 | 74.8 |
| household-chores | 26.2 | 38.7 | 42.1 | 45.7 | 57.7 |
| vehicle | 55.7 | 50.1 | 54.7 | 55.7 | 47.5 |
| walking | 32.0 | 37.6 | 36.4 | 41.3 | 34.7 |
| bicycling | 40.0 | 55.2 | 64.0 | 70.0 | 66.7 |
| sports | 24.9 | 27.4 | 26.6 | 19.4 | 31.6 |
| mixed-activity | 4.8 | 5.4 | 9.7 | 6.6 | 13.3 |
| standing | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| manual-work | 1.3 | 0.0 | 2.8 | 2.9 | 0.0 |
| **Macro-F1** | **32.9** | **36.8** | **39.0** | **40.2** | **41.4** |
| **Balanced Accuracy** | **33.1** | **35.7** | **39.8** | **39.4** | **42.4** |

Table 9: Evaluation metrics across context lengths. At 15 min, ITFormer encountered out-of-memory errors (†).

| | | **OpenTSLM-Flamingo** | | **ITFormer** | |
|---|---|---|---|---|---|
| **Duration** | **Timesteps** | **Macro-F1** | **Bal. Acc.** | **Macro-F1** | **Bal. Acc.** |
| 2.56 s | 128 | 32.9% | 33.1% | 13.9% | 17.8% |
| 10 s | 500 | 36.8% | 35.7% | 21.8% | 25.4% |
| 60 s | 3,000 | 39.0% | 39.8% | 21.6% | 25.0% |
| 5 min | 15,000 | 40.2% | 39.4% | 15.8% | 17.3% |
| 15 min | 45,000 | 41.4% | 42.4% | OOM† | |

However, as noted by the authors, most datasets found operate on a fixed-length windows of $T{=}512$ timesteps and focus on holistic tasks that do not require localizing or reasoning about specific events within extended temporal contexts. TS-Haystack complements this line of work, testing whether models preserve fine-grained temporal information across contexts spanning orders of magnitude beyond typical training windows.

**Time Series Understanding Benchmarks.** TimeSeriesExam (Cai et al., 2024) evaluates LLMs across five core time series understanding categories: pattern recognition, noise understanding, similarity analysis, anomaly detection, and causality analysis. While their procedural generation approach enables scalability, the benchmark relies on *synthetic* time series and focuses primarily on statistical properties (e.g., "Do the two time series have similar variance?", "What is the trend?"). In contrast, TS-Haystack emphasizes *semantic interpretation* using real sensor data with activity labels, testing whether models can derive real-world meaning from temporal patterns rather than merely describing statistical features.

Time-MQA (Kong et al., 2025) is more closely aligned with our objective, unifying forecasting, imputation, anomaly detection, classification, and open-ended reasoning under a question-answering framework. Their TSQA dataset comprises approximately 200k question-answer pairs across diverse domains. However, Time-MQA does not specifically probe *long-context retrieval*, the ability to locate and reason about specific events within extended temporal contexts.

**Long-Context Evaluation in Other Modalities.** The needle-in-a-haystack paradigm originated in LLM evaluation (Kamradt, 2023), where a specific fact is embedded within long text documents to test retrieval accuracy. BABILong (Kuratov et al., 2024) extends this to reasoning tasks, demonstrating that popular LLMs effectively utilize only 10–20% of their claimed context length. For vision-language models, MMNeedle (Wang et al., 2024) adapts the paradigm to multimodal retrieval, stress-testing models' ability to locate target sub-images within large visual contexts. To our knowledge, TS-Haystack is the first benchmark to apply the needle-in-a-haystack methodology to continuous time series data, where the challenge is compounded by the absence of discrete token boundaries and the need to preserve temporal precision under aggressive compression.

**Distinguishing Contributions.** Unlike synthetic benchmarks (Cai et al., 2024), TS-Haystack uses *real* accelerometer recordings with ecologically valid activity patterns. Our statistically validated needle insertion protocol (Appendix B.3) ensures that inserted activities are indistinguishable from natural occurrences, enabling scalable semi-synthetic dataset generation without introducing detectable artifacts. This methodology may prove valuable beyond benchmarking, for instance, in

Table 10: Task configuration parameters for TS-Haystack generation. Needle length is specified as a percentage of context length. All tasks use 2% margin and minimum gap ratios (capped at 100 samples for longer contexts).

| Task | Needle Length | Background | Task-Specific |
|---|---|---|---|
| Existence | 2–10% | any | 1 – 3 distractors |
| Localization | 2–10% | any | 1 – 3 distractors |
| Counting | 2–8% | any | 1 – 5 bouts |
| Ordering | 2–10% | any | – |
| State Query | 1–5% | mixed | 2 – 5 states, min 20% duration |
| Antecedent | 2–8% | any | 10-sample adjacency gap |
| Comparison | 2–8% | any | 2 – 4 bouts, min 2% duration diff |
| Multi-Hop | 2–6% | any | $K$ position index$\sim [0.4, 0.4, 0.2]$ |
| Anomaly Detection | 3–15% | pure | 1 – 3 distractors |
| Anomaly Localization | 3–15% | pure | 1 – 3 distractors |

augmenting training data for domains where anomalies are scarce, such as rare event detection in healthcare or industrial monitoring.

### B.2 Dataset Construction

**Pre-processing.** For each participant in Capture24, we construct an activity timeline by merging consecutive annotations of the same activity into *bouts*, recording start time, end time, and duration. We aggregate all bouts across participants into a *bout index* organized by activity class, enabling efficient sampling of needle candidates by activity type and duration.

**Needle selection.** We deliberately avoid sampling according to empirical transition probabilities: real activity sequences exhibit strong temporal dependencies that standard HAR pipelines exploit via HMM smoothing (Chan et al., 2024). Uniform random needle-background pairings ensure models cannot game the benchmark through learned Markovian priors, requiring genuine temporal retrieval for correct answers.

**Needle insertion.** To ensure inserted needles are statistically consistent with the surrounding context, we apply per-channel mean alignment:

$$\mathbf{x}' = \mathbf{x}_{\text{needle}} - \boldsymbol{\mu}_{\text{needle}} + \boldsymbol{\mu}_{\text{target}} \tag{1}$$

where $\boldsymbol{\mu}$ denotes the per-channel mean. This adjusts for participant-specific sensor biases (orientation, placement) while preserving the activity's characteristic amplitude and temporal dynamics. To ensure smooth transitions at insertion boundaries, we apply cosine blending over a variable window of $w$ samples, weighting the needle contribution as $\alpha(t) = \frac{1}{2}(1 - \cos(\pi t/w))$ to gradually interpolate between background and needle signals.

**Question generation.** Questions are generated from 20 manually-crafted templates per task, with activity names and timestamps filled programmatically per sample. Contextual timestamps are included in the input prompts and grounded in real clock time (e.g., "02:34:56:789 AM") derived from the original Capture24 recording metadata, enabling naturalistic queries aligned with real-world applications. All experiments use a fixed master seed with deterministic sample generation for reproducibility.

**Configurable difficulty.** TS-Haystack supports configurable difficulty along multiple axes: needle position (beginning/middle/end/random), needle duration, minimum gap between bouts, distractor density, and other task-specific parameters. For our main dataset generation, we use random needle positions with the task-specific settings shown in Table 10.

**Chain-of-thought annotations.** For each sample, we generate chain-of-thought (CoT) rationales using ChatGPT 4.1 mini (OpenAI, 2023), conditioned on image plots of the accelerometer data along with rich metadata including activity timelines, bout boundaries, and signal statistics. These annotations support training and evaluation of reasoning capabilities beyond direct answer extraction.

### B.3 Insertion Quality Validation

To verify that our style transfer and cosine blending protocol produces clean insertions without detectable artifacts, we design a classifier-based validation test. The key insight is that if a machine learning model cannot distinguish samples with insertions from samples without, the insertion process introduces no systematic artifacts.

**Methodology.** We construct a binary classification task:

- **Class 0 (negative):** Pure background windows with no insertion.
- **Class 1 (positive):** Pure background windows with a same-activity needle inserted from a *different participant*.

Using same-activity needles isolates insertion artifacts from activity-class differences. If the classifier succeeds, it must be detecting blending imperfections rather than activity signatures.

We generate 5,000 training samples and 500 test samples at 2 and 3 seconds context length (200 and 300 samples at 100Hz), with needle lengths ranging from 2–8% of context aligned with the true dataset configurations and balanced class distributions. An XGBoost classifier (Chen & Guestrin, 2016) is trained on flattened triaxial accelerometer features (600 and 900 dimensions) using 100 estimators with max depth 6.

**Results.** The classifier achieves an AUC of 0.499 and 0.490 respectively for different context lengths, close to random chance (0.50). This confirms that our mean-shift normalization combined with cosine blending produces insertions that are statistically undetectable, even to a gradient-boosted ensemble operating on raw signal features. The protocol successfully preserves activity dynamics while eliminating participant-specific biases and boundary discontinuities.

### B.4 ITFormer Extended Discussion

Several limitations of our ITFormer evaluation warrant discussion. First, we did not perform hyperparameter tuning specific to the Capture24 task or TS-Haystack tasks; the learning rate, batch size, and training duration were adopted from the original work and may be suboptimal for accelerometer-based activity recognition. Second, the LLM backbone was kept frozen: finetuning or using LoRA adapters could improve the model's ability to interpret time series prefix embeddings. Finally, OOM errors limited the model from learning representations from longer context lenghts, effectively training on a smaller dataset than OpenTSLM-Flamingo. Further ablation experiments on patching strategies could alleviate memory constraints or preserve timeseries features of the specific dataset. Future work could investigate these directions to determine whether the performance gap with latent resampling methods narrows under a more optimized full-resolution setup.

### B.5 Oracle Model Experiment

To further isolate time series representation as the bottleneck for TSLM performance on our benchmark, we introduce an Oracle-based upper bound evaluation. The oracle model receives the ground-truth activity timeline as text prepended to the standard prompt as shown in Table 11, bypassing the requirement for the time series encoder to provide the activity recognition and temporal grounding information in its representations for successful answering.

Table 11: Oracle prompt for an example multi-hop task. *Italic text* indicates the oracle-only addition; regular text is shared across configurations.

**Prompt (*Pre-prompt Oracle* + Regular)**

*Activity Timeline (Ground Truth):*
*Recording: 10:53:45 PM – 10:53:55 PM*
*10:53:45 – 10:53:50: sleep*
*10:53:50 – 10:53:50: sitting [inserted]*
*10:53:50 – 10:53:51: sleep*
*10:53:51 – 10:53:51: sitting [inserted]*
*10:53:51 – 10:53:51: sleep*
*10:53:51 – 10:53:52: mixed-act. [inserted]*
*10:53:52 – 10:53:55: sleep*

You are given accelerometer data in all three dimensions from a wrist-worn sensor. The recording spans from 10:53:45 PM to 10:53:55 PM.

Question: When was the second sitting activity before mixed-activity recorded?

[**accelerometer data x/y/z axes**]

Instructions: Analyze the accelerometer data carefully. Think step-by-step about what the signal patterns indicate. Write your reasoning as a natural paragraph. End your response with "Answer: <your answer>"

### B.6 TASK DETAILS

In the following, we provide detailed descriptions with visual examples sampled from our sample generation pipeline for each task. Task variations such as *non* Existence, *non* Anomaly Detection or all 8 Comparison variants are excluded for brevity.

**Task 1: Existence.** *"Is there {activity} in this recording?"*
This task tests whether the model can detect the presence of a specific activity pattern within a longer recording. To prevent variance-based detection shortcuts (where models detect *any* insertion rather than the specific activity), we insert multiple same-regime distractors. The model must learn activity-specific patterns rather than relying on statistical anomalies.
**Construction:** (1) Sample background window; (2) Select activity regime (sedentary/active); (3) Insert $N$ needles from regime activities not in background; (4) For positive samples, ask about an inserted activity; for negative, ask about a non-inserted activity from the same regime.
**Example:** *Q: "Is there manual-work in this recording?" A: "No."*

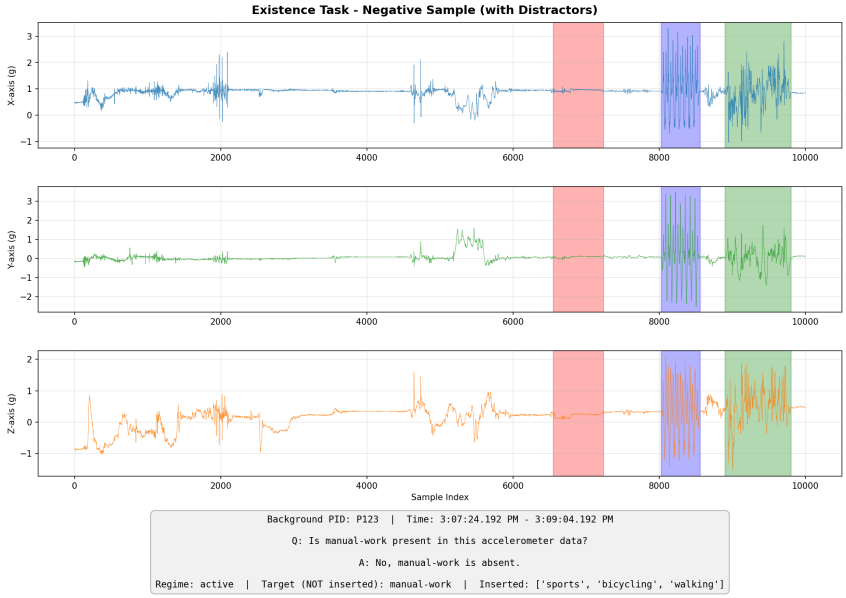



Figure 3: Task 1: Existence. The model must detect whether a specific activity is present among same-regime distractors.

**Task 2: Localization.** *"When did the {activity} bout occur?"*
Extends existence by requiring temporal grounding. The model must not only detect the activity but report its precise time range. Distractors with similar statistical properties force the model to identify activity-specific temporal signatures.
**Construction:** Same as Existence, but answer requires the timestamp range of the target needle.
**Example:** *Q: "When did the sports bout occur?" A: "From 02:09:52:635 to 02:10:08:967."*

**Task 3: Counting.** *"How many {activity} bouts occurred?"*
Tests enumeration capabilities: the model must detect all instances of an activity and count them correctly. Minimum gaps between bouts ensure distinguishability while making the task non-trivial.
**Construction:** (1) Sample background; (2) Sample $N \in [1, 5]$ bouts of target activity; (3) Insert at non-overlapping positions with minimum gap constraint.
**Example:** *Q: "How many standing bouts occurred?" A: "3."*

**Task 4: Ordering.** *"Did {activity_a} occur before {activity_b}?"*
Tests temporal comparison between two distinct activities. We use needle insertion rather than natural bout selection to prevent models from exploiting learned activity transition statistics. Using random needle sampling pairings instead of existing activity transitions in the data ensure models cannot game the benchmark through learned Markovian priors, requiring genuine temporal retrieval for correct answers (Chan et al., 2024).
**Construction:** (1) Sample two distinct activities $A$ and $B$; (2) Sample background excluding both; (3) Randomly assign temporal order (50/50 balance); (4) Insert both needles sequentially.
**Example:** *Q: "Did sports occur before sleep?" A: "Yes."*

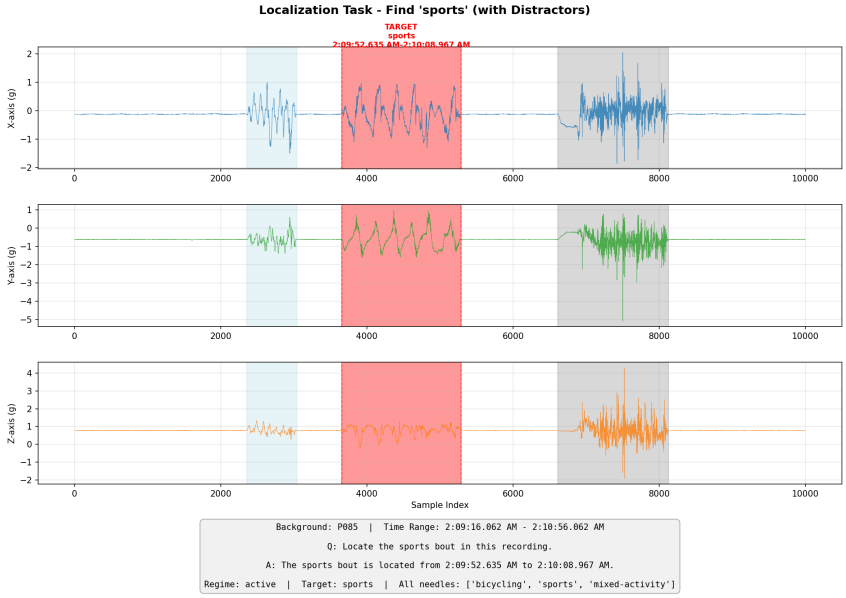



Figure 4: Task 2 (Localization). The model must identify and temporally ground a specific activity

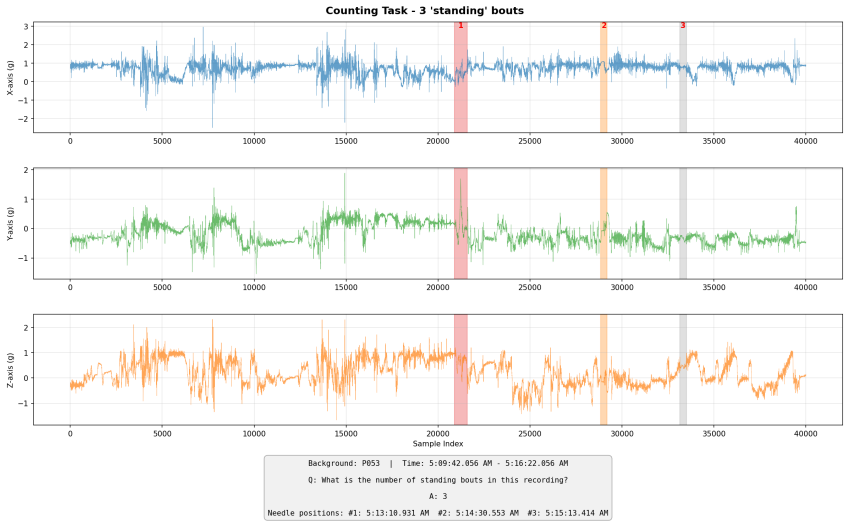



Figure 5: Task 3 (Counting). The model must enumerate all occurrences of a specific activity

**Task 5: State Query.** *"What was the activity level when {event} occurred?"*
Tests cross-scale integration: the model must detect a local event (needle, seconds-scale) and identify the surrounding global activity regime (minutes-scale). This requires understanding hierarchical temporal context.
**Construction:** (1) Sample mixed background with 2–5 activity states; (2) Select a global state region; (3) Insert needle within that region; (4) Ask about the global state, not the needle activity.
**Example:** *Q: "What was the overall activity when the vehicle activity occurred?" A: "Standing."*

**Task 6: Antecedent.** *"What activity occurred immediately before {target}?"*
Tests sequential reasoning: the model must locate the target activity and identify what directly preceded it. Two-needle insertion gives control over the antecedent-target pairing, preventing reliance on natural transition statistics.
**Construction:** (1) Sample background (prefer low-activity); (2) Sample antecedent $A$ and target $T$; (3) Insert $A$ followed immediately by $T$ with small adjacency gap.
**Example:** *Q: "Looking at the sequence, what came before bicycling?" A: "Sports."*

**Task 7: Comparison.** *"What was the {longest/shortest} period {with/without} {activity}?"*

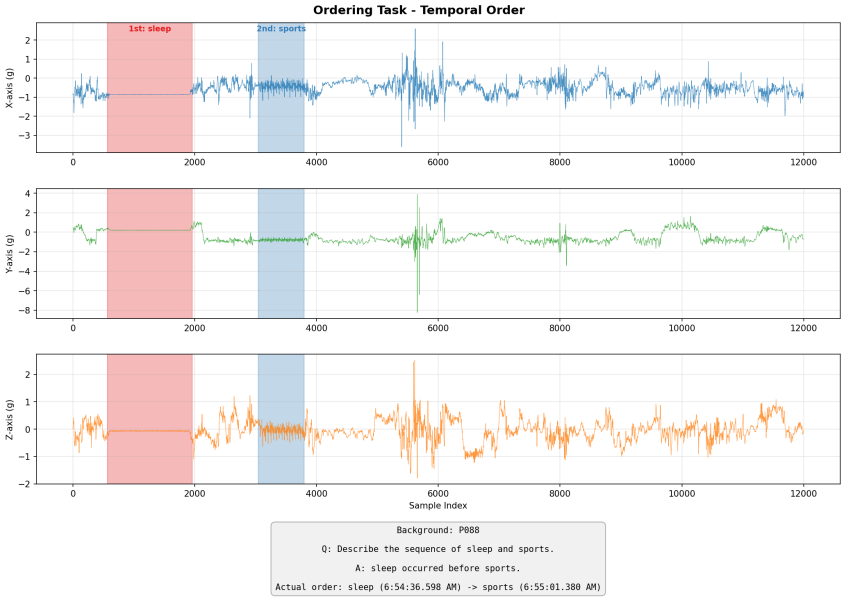



Figure 6: Task 4 (Ordering). The model must determine the temporal relationship between two activities


Figure 7: Task 5 (State Query). The model must identify the global activity regime surrounding a local event

Tests comparison reasoning with four question variants: longest/shortest $\times$ with/without. The "without" polarity requires understanding negation and computing gaps between activity bouts. All inserted bouts have distinct durations to ensure unambiguous answers.
**Construction:** (1) Sample question type (extremum $\times$ polarity); (2) Insert $N \geq 2$ bouts with distinct durations; (3) For "with": answer is extremum bout; for "without": answer is extremum gap.
**Example:** *Q: "What was the longest period without household-chores?" A: "From 8:32:52:619 to 08:35:30:682."*

**Task 8: Multi-Hop.** *"When did the K-th {target} occur {before/after} {anchor}?"*
Tests multi-step reasoning: (1) locate the anchor activity, (2) identify target bouts relative to anchor, (3) count to the $K$-th occurrence in the specified direction. Optional distractors on the opposite side test directional understanding.

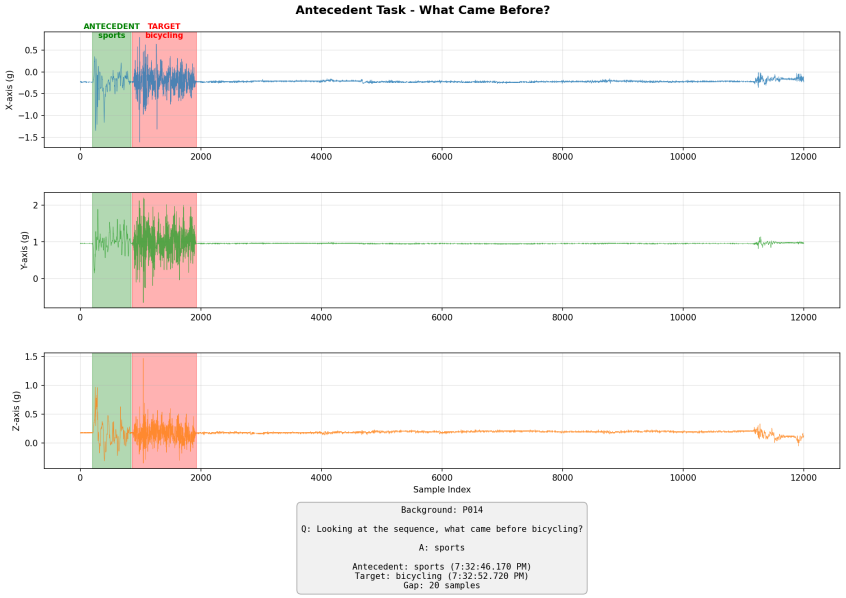



Figure 8: Task 6 (Antecedent). The model must identify the activity immediately preceding a target

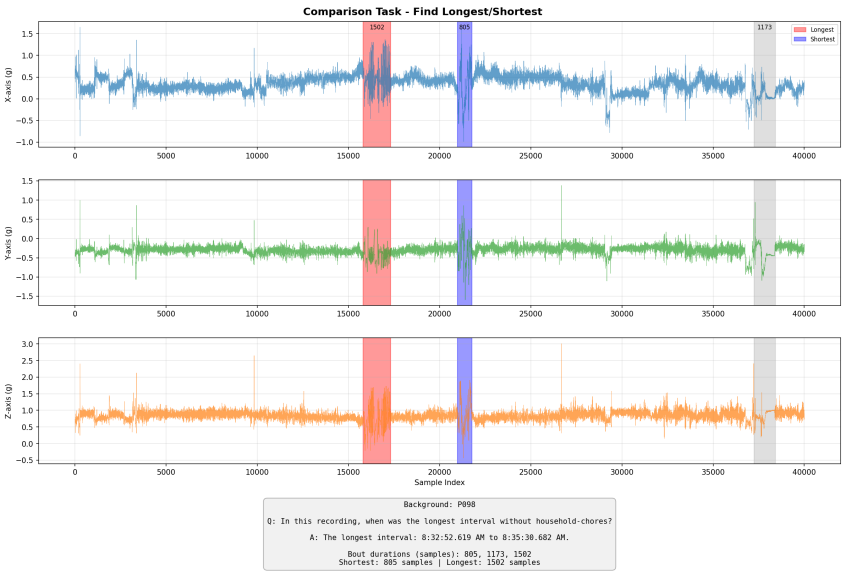



Figure 9: Task 7 (Comparison). The model must compare durations across multiple activity bouts or gaps

**Construction:** (1) Sample anchor $A$ and target $T$; (2) Sample $K \in \{1, 2, 3\}$ and direction; (3) Position anchor with room for $K$ targets; (4) Insert $K$ target needles in order.

**Example:** *Q: "When did the third sports bout occur after mixed-activity?" A: "From 09:38:32:874 to 09:38:40:924."*

**Task 9: Anomaly Detection.** *"Is there an anomaly in this recording?"*

Tests contextual reasoning: the model must detect activity that is anomalous *relative to the background regime*. Unlike Existence (where the target is given), the model must identify what constitutes an anomaly. We define anomalies as cross-regime insertions (e.g., vigorous activity in a sedentary background). We partition activities into two regimes: *sedentary* (sleep, sitting, standing, vehicle) and *active* (walking, mixed-activity, bicycling, manual-work, sports, household-chores). Mandatory same-regime distractors prevent "insertion detected" shortcuts: both positive and negative samples contain insertions, so the model must identify regime mismatch.

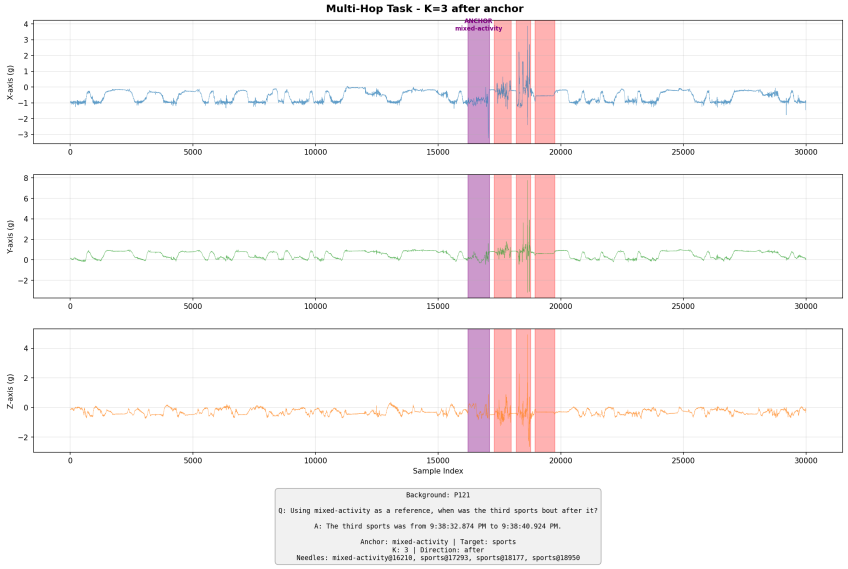



Figure 10: Task 8 (Multi-Hop). The model must locate an anchor, then count to the $K$-th target in a specified direction

**Construction:** (1) Sample pure background from one regime; (2) Positive: insert 1 cross-regime needle (anomaly) + $N$ same-regime distractors; Negative: insert only same-regime distractors.
**Example:** *Q: "Is there an anomaly in this recording?" A: "Yes, there is anomalous sitting activity in the active background."*


Figure 11: Task 9 (Anomaly Detection). The model must identify cross-regime activity without being told what to look for

**Task 10: Anomaly Localization.** *"Is there an anomaly, and if so, when did it occur?"*
Extends anomaly detection by requiring temporal localization. The model must detect the cross-regime violation and specify its time range. This combines contextual reasoning with precise temporal grounding.
**Construction:** Same as Anomaly Detection, but positive answers must include the time range of the anomalous activity.
**Example:** *Q: "Is there an anomaly, and if so, when?" A: "Yes, there is anomalous sitting activity from 04:21:20:792 to 04:21:34:333."*

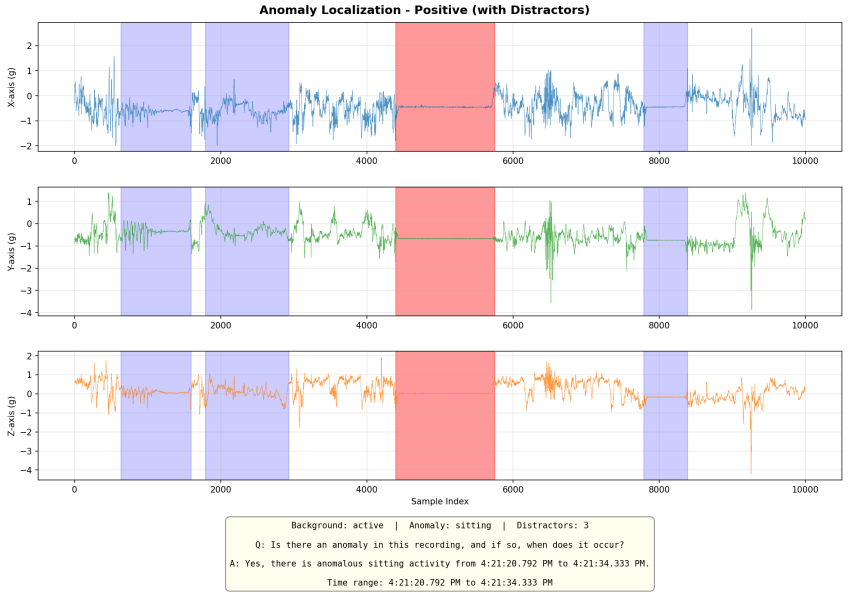



Figure 12: Task 10 (Anomaly Localization). The model must detect and temporally ground cross-regime anomalies