LAPRAS: LATENT REASONING FOR TIME SERIES LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Time Series Language Models (TSLMs) offer a promising path toward time series understanding by reasoning over temporal signals and producing natural language answers and explanations. A common approach is Chain-of-Thought (CoT), which generates step-by-step rationales describing relevant signal patterns and connecting them to final answers. Although these models learn from reference CoT traces during post-training, generating faithful descriptions of input time series at inference time remains challenging. The model is required to express high-dimensional and continuous temporal representations in discrete language tokens, which may lead to neglecting task-relevant patterns or describing them inaccurately. Because subsequent reasoning steps build on these descriptions, early errors can propagate through the reasoning chain, ultimately leading to incorrect answers accompanied by plausible explanations that are inconsistent with the input signal. To address this, we propose ***Lapras*** *(**La**tent **P**ost-trained **R**easoning **A**cross **S**eries)*, a post-training framework that equips TSLMs with latent reasoning. A model trained with Lapras reasons through a sequence of *continuous thoughts* in the joint time series–language space, producing text only for the final answer. It learns this through teacher-student self-distillation, where a teacher trained on CoT reference traces reasons explicitly through text. The student is trained to align its hidden states with the teacher's at the answer stage, transferring the teacher's reasoning ability into the student's latent computation. We evaluate Lapras across four TSLM backbones on five diverse time series question answering benchmarks. Lapras improves average F1 by up to 10.79% over explicit CoT while generating $23.9\times$ fewer tokens. Importantly, Lapras's generated continuous thoughts can be decoded into readable reasoning traces through standard language decoding, preserving the capability to provide textual explanations. Together, these results highlight Lapras as a promising post-training paradigm for TSLMs that enables efficient and effective reasoning at inference while supporting interpretability.

1 INTRODUCTION

Time Series Language Models (TSLMs) treat time series as a native modality for LLMs, enabling reasoning over sensor signals in natural language across domains ranging from wearable health to industrial monitoring (Xie et al., 2025; Wang et al., 2025b). To develop reasoning capability, existing works mainly elicit reasoning through Chain-of-Thought (CoT) supervised finetuning (SFT), a common post-training method that further finetunes pretrained models on reference CoT traces (Langer et al., 2026). After training, the model generates its own step-by-step rationale before producing the answer, providing a textual explanation that users can inspect against the underlying signal. For example, for the ECG question answering task, a TSLM can generate rationales describing observed waveforms such as an irregular R-R interval and explain how these are linked to a diagnostic answer like candidate arrhythmia (Xu et al., 2026).

However, learning from reference CoT traces during post-training does not ensure that models generate traces faithful to the input time series at inference (Turpin et al., 2023). Generating a CoT requires the model to express high-dimensional, continuous time series representations as discrete language tokens at each step. The resulting description may neglect temporal patterns or characterize them inaccurately. Because subsequent reasoning steps are predominantly conditioned on this description rather than on the original continuous representation, early omissions or errors can

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.

propagate and compound throughout the reasoning chain, ultimately leading to an incorrect answer (Zhang et al., 2023). For instance, for a TSR question over a temperature signal, a generated CoT trace may correctly identify an upward trend and a peak but overlook that the increase is temporary. Subsequent reasoning may then interpret this temporary increase as a sustained abnormal event and arrive at an incorrect operational decision, such as activating emergency backups when standard capacity is sufficient (Figure 9). We term this limitation the *verbalization bottleneck*: generated textual descriptions may omit or misrepresent task-relevant temporal patterns, and these errors can propagate through subsequent reasoning. Our analysis in Section 3.1 provides empirical support for this bottleneck, showing that a substantial gap in answer accuracy emerges between predicted and reference CoT traces during signal description and persists through subsequent reasoning.

Recent latent reasoning frameworks offer a potential way to address this bottleneck by reasoning in continuous hidden states rather than generating intermediate text (Hao et al., 2025; Shen et al., 2025). At each reasoning step, the model's hidden state is passed directly as input to the next step, replacing the text tokens used in CoT. Although these methods have demonstrated promising results on text-based reasoning tasks, where both inputs and reasoning traces share the same discrete token space, their effectiveness for reasoning over the continuous time series representation space remains underexplored. Consequently, we ask: *can TSLMs reason more effectively in a continuous latent space than through discrete tokens?*

Motivated by this question, we propose ***Lapras*** *(**La**tent **P**ost-trained **R**easoning **A**cross **S**eries)*, a post-training framework for latent reasoning in TSLMs. Specifically, we adopt the self-distillation framework (Shen et al., 2025), which lets the model learn from the reasoning signal in explicit CoT traces during training without having to verbalize numerical patterns as text at inference. When training with Lapras, the teacher processes the time series, question, and explicit CoT reference traces, while the student reasons over the same time series and question through continuous thoughts. The student learns to match the teacher's hidden states at the answer position, without requiring each continuous thought to generate a textual reasoning step. At inference, the student performs intermediate reasoning in the joint time series–language space and generates text only for the final answer, mitigating the verbalization bottleneck by avoiding intermediate textual descriptions of the signal. Our experiments compare Lapras against existing reasoning approaches across different TSLM backbones and demonstrate that Lapras improves average answer accuracy over explicit CoT while substantially reducing token generation. In summary, our key contributions are:

- We identify and empirically characterize the *verbalization bottleneck* in TSLMs, showing that prediction inaccuracy for explicit CoT emerges during signal description and persists through reasoning chains (Section 3.1).
- To address this limitation, we propose Lapras, which adapts teacher-student self-distillation to post-train TSLMs for latent reasoning, letting the model learn from reference CoT while performing intermediate reasoning through continuous thoughts (Section 3.2).
- Across four TSLM backbones and five reasoning benchmarks, Lapras improves average F1 by up to 10.79% over CoT while substantially reducing the number of generated tokens. Qualitative analyses show that continuous thoughts of Lapras can be decoded into readable reasoning traces, retaining explanation capability (Section 5).

2 Related Work

Time Series Language Models. While Time Series Foundation Models (TSFMs) excel at forecasting and fixed-label classification, their rigid outputs preclude open-ended reasoning over sensor signals (Ansari et al., 2024; Das et al., 2024; Goswami et al., 2024; Luo et al., 2025). Recent works aim to bridge this gap, progressing from serializing raw data as text (Liu et al., 2023; Kim et al., 2024; Gruver et al., 2024) to encoding signals into continuous representations that the LLM processes directly (Xie et al., 2025; Langer et al., 2026; Chen et al., 2026). These models elicit reasoning via explicit CoT through supervised finetuning (Langer et al., 2026; Messica et al., 2026; Guan et al., 2026) or reinforcement learning (Messica et al., 2026; Guan et al., 2026). However, verbalizing intermediate rationales into discrete tokens may inflate inference latency (Wu et al., 2025) and risk hallucination due to excessive reliance on linguistic priors (Zhu et al., 2026). In contrast, Lapras conducts multi-step reasoning entirely within the model's continuous latent space, sidestepping intermediate token generation while still decoding final answers into free-form text.

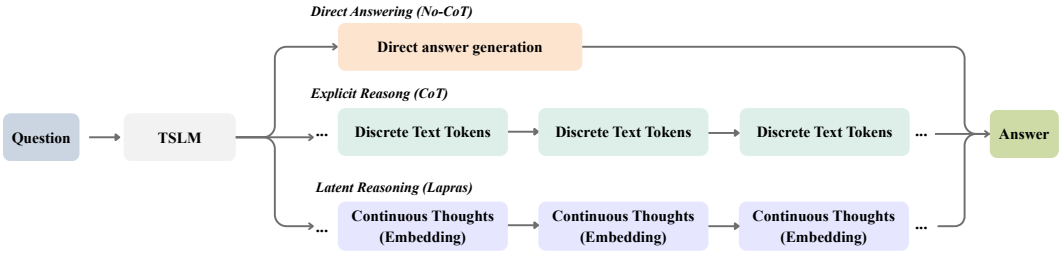


Figure 1: **Three reasoning paradigms for TSLMs.** *(i) Direct answering (No-CoT)* maps the input directly to a textual answer without generating intermediate reasoning steps. *(ii) Explicit reasoning (CoT)* expresses intermediate reasoning as discrete text tokens and conditions on these steps to produce the answer. *(iii) Latent reasoning (Lapras)* performs intermediate reasoning through continuous thoughts in latent space and decodes text only for the final answer.

Implicit Chain-of-Thought Reasoning. Implicit reasoning methods replace discrete decoded tokens with continuous hidden states which are fed back as input embeddings, freeing models from enforcing textual coherence or committing prematurely to a single reasoning path (Gao et al., 2025; Hao et al., 2025). Rather than verbalizing steps, iCoT (Gao et al., 2025) progressively removes CoT tokens during training to internalize the reasoning process directly into model weights. COCONUT (Hao et al., 2025) preserves these intermediate positions by feeding the last-layer hidden state back as the next input embedding, which serves as a continuous thought. CODI (Shen et al., 2025) adopts these continuous thoughts and supervises them through self-distillation against an explicit CoT teacher. While these methods do not yet consistently match explicit CoT in accuracy, eliminating intermediate token generation substantially reduces inference latency, establishing latent reasoning as a compelling alternative paradigm. In parallel, recent vision-language models interleave latent visual tokens with text tokens to manipulate visual evidence without generating explicit images (Asadi et al., 2026; Li et al., 2025a; Wang et al., 2025a). However, unlike static image patches that carry independent semantic meaning, time series signals may require jointly interpreting transient local events (e.g., peaks) and global properties (e.g., periodicity and cross-channel dependencies) that are inherently difficult to express in natural language. Lapras, instead, reasons in a joint time series–language space, directly coupling temporal and spatial dynamics with semantic anticipation.

3 METHOD

3.1 DIAGNOSING THE VERBALIZATION BOTTLENECK

In time series question answering, a TSLM takes a time series and a question as input and generates a textual answer, either directly or through intermediate reasoning steps (Figure 1). Explicit CoT requires a TSLM to describe temporal evidence in language and condition subsequent reasoning on these descriptions. Inaccurate or incomplete descriptions may therefore affect the final answer (i.e., *verbalization bottleneck*). To investigate this issue, we examine how answer accuracy evolves as the model receives successive steps of predicted or reference CoT traces. Specifically, we investigate whether an accuracy gap emerges during signal description and whether subsequent reasoning closes this gap.

Setup. We adapt the *early answering* protocol from Lanham et al. (2023), which tests how a model's final answer depends on its predicted CoT traces. We apply it to SLIP-1B finetuned with CoT-SFT. Specifically, we progressively truncate both predicted and reference reasoning traces one sentence (hereafter, a *step*) at a time, prompting the model for a final answer after each step to track accuracy. Based on manual inspection, we assign each sentence of the CoT to one of two stages: *describe*, where it summarizes the input signal in words, and *reason*, where it draws conclusions from the preceding description. These annotations allow us to examine when answer accuracy diverges between the two conditions and how the gap changes through subsequent reasoning.

Results and Implications. As shown in Figure 2, a substantial performance gap emerges in the *describe* stage, reaching $\Delta 0.22$ on TSR and $\Delta 0.17$ on Sleep. This gap persists through the final *reason* stage, where the predicted trace's accuracy plateaus. More results are provided in Figure 10. These findings provide empirical support for the *verbalization bottleneck* where translating continuous time series into text during the describe stage discards task-relevant evidence, introducing errors that propagate through the reasoning chain. Moreover, this analysis offers a possible explanation for why CoT-SFT struggles to outperform No-CoT baselines (Table 1). Ultimately, this motivates learning from reference rationales while reasoning in continuous space, eliminating the need to generate text descriptions at inference.

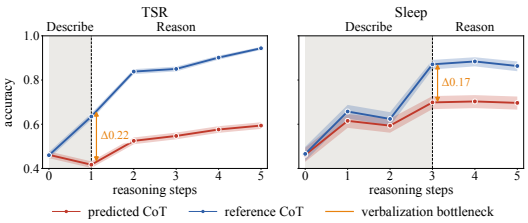


Figure 2: **Early answering with predicted and reference CoT.** A substantial accuracy gap between predicted and reference traces is evident by the end of signal description and persists through subsequent reasoning. Mean and standard deviation are computed over the full test set of each benchmark.

3.2 LAPRAS

Overview. To address the verbalization bottleneck, we introduce Lapras (Latent Post-trained Reasoning Across Series), which reasons over the question and time series through continuous thoughts in latent space, decoding text only for the final answer. As shown in Figure 3, Lapras adopts a knowledge distillation framework (Caron et al., 2021; Shen et al., 2025). Here, a teacher network is trained on CoT reasoning traces, while a student network learns to match the teacher's hidden states through latent reasoning. Both the student and teacher models share the same TSLM architecture and parameters. Formally, given a multivariate sensor series X_s and a text prompt X_q (comprising context and a question), a TSLM generates a textual answer X_a. We denote the time series representation by H_s and the prompt embeddings by $H_q = \mathrm{embed}(X_q)$. The language model f_ϕ conditions on both inputs, integrating H_s through the backbone's existing architecture, and returns the last hidden state at the final position. The answer is decoded autoregressively as $X_a \sim p_\phi(\cdot \mid H_s, H_q)$.

Teacher-Student Self-distillation. The teacher and student share the same TSLM architecture but perform intermediate reasoning differently. Given the time series and question, the teacher is trained with standard CoT-SFT: it processes the reference CoT trace and is trained with token-level cross-entropy over the traces and answers, $\mathcal{L}_{\mathrm{CE}}^{\mathrm{teacher}}$. The student instead generates continuous thoughts and is trained with cross-entropy over only the answer tokens, $\mathcal{L}_{\mathrm{ans}}^{\mathrm{student}}$. To transfer information from the reference rationale to the student's latent reasoning, we align the hidden states used to predict the first answer token.

Lapras introduces two learned markers, $\langle \mathrm{bot} \rangle$ (*begin-of-thought*) and $\langle \mathrm{eot} \rangle$ (*end-of-thought*), which delimit the latent reasoning process. Given a prompt X_q and a series X_s, the student embeds both and appends $\langle \mathrm{bot} \rangle$, which puts it in *latent mode*,

$$
\begin{aligned}
&S_0 = [\, H_q,\, H_s,\, \langle \mathrm{bot} \rangle \,], \\
&z_k = f_\phi(S_{k-1}), \qquad\qquad S_k = [\, S_{k-1};\, \pi(z_k) \,], \\
&X_a \sim p_\phi(\cdot \mid [\, S_K;\, \langle \mathrm{eot} \rangle \,]),
\end{aligned}
\tag{1}
$$

where π is a learned projection from the last hidden state back into the input embedding space and $k = 1, \ldots, K$. In latent mode the student never samples a token. Instead, after each forward pass, it extracts the last hidden state z_k at the final position and projects it back into the input embedding space via a small learned MLP π. The projected vector is then appended to the input sequence as if it were a new token embedding, and the model runs another forward pass. We refer to $z_1, \ldots, z_K$ as *continuous thoughts* (Hao et al., 2025; Shen et al., 2025): intermediate reasoning representations that encode the model's reasoning process without being decoded into language. After generating $K = 6$ continuous thoughts, the student returns to *language mode* at $\langle \mathrm{eot} \rangle$ to decode the final answer X_a in language tokens. X_a is supervised with $\mathcal{L}_{\mathrm{ans}}^{\mathrm{student}}$, the cross-entropy over the answer tokens.

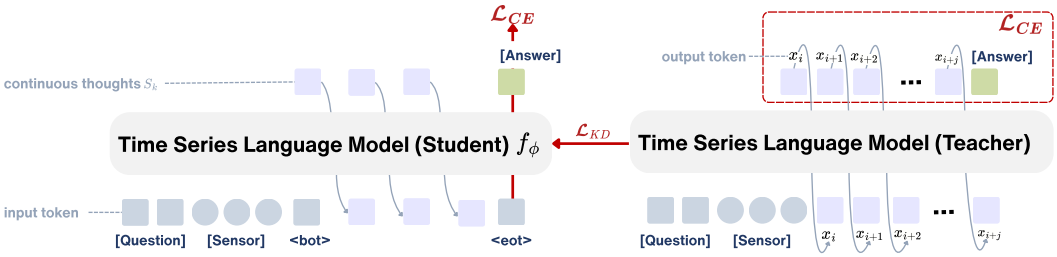


Figure 3: **Training framework of Lapras.** The student (left) produces K continuous thoughts between $\langle \text{bot} \rangle$ and $\langle \text{eot} \rangle$, feeding each hidden state back as the next input embedding without emitting any token, and predicts the answer under $\mathcal{L}_{\text{ans}}^{\text{student}}$. The teacher (right) writes a full CoT (e.g., $[x_i, x_{i+1}, \ldots, x_{i+j}]$) and predicts the answer under $\mathcal{L}_{\text{CE}}^{\text{teacher}}$. A distillation loss $\mathcal{L}_{\text{KD}}$ aligns their hidden states at the anchor position (the first answer token). The teacher is discarded at inference.

Training & Inference. Student and teacher are trained jointly, and the distillation couples them at the *anchor*, the first answer token of each, where we align their hidden states over all layers,

$$\mathcal{L}_{\text{KD}} = \frac{1}{M+1} \sum_{\ell=0}^{M} \left| \text{sg}\left[h_{\text{teacher}}^{\ell}\right] - h_{\text{student}}^{\ell} \right|, \tag{2}$$

where M is the number of transformer layers, h^{ℓ} is the hidden activation of the ℓ-th layer of the TSLM, and sg denotes the stop-gradient operation. Combining this term with the two cross entropies gives the full objective $\mathcal{L} = \mathcal{L}_{\text{ans}}^{\text{student}} + \lambda \mathcal{L}_{\text{KD}} + \beta \mathcal{L}_{\text{CE}}^{\text{teacher}}$, which supervises the student both on the answer it produces and on the state it passes through on the way to that answer. The distillation weight λ sets how strongly the student is pulled toward the teacher's states relative to its own answer loss, and we set it to $\lambda = 10$ by default and adjust it per task as reported in Table 7. The teacher weight β scales the teacher's own cross entropy, and we keep it at $\beta = 1$ throughout.

During training, we apply a standard input regularizer that masks a fraction (ρ=0.3) of the sensor patches (Das et al., 2024). Our ablation study shows that this improves performance by 1.80% on average (see Table 3). At inference, we discard the teacher branch and run only the student network (Figure 3, left) through the following steps. First, the student ingests the prompt X_q and sensor series X_s. Second, it performs K latent reasoning steps as described in Equation 1, emitting no intermediate tokens. Finally, after appending $\langle \text{eot} \rangle$, the student transitions back to language generation and decodes the final answer X_a conditioned on $[\, S_K; \langle \text{eot} \rangle \,]$.

4 Experiment

Datasets. We evaluate Lapras on five time series question answering tasks: ECG (Oh et al., 2026) (cardiological diagnosis), Engine (Wang et al., 2025b) (aero-engine fault diagnosis), HAR (Langer et al., 2026) (human activity recognition), Sleep (Langer et al., 2026) (sleep-stage classification), and TSR (Merrill et al., 2024) (counterfactual prediction).

To support explicit CoT post-training, we construct reference reasoning traces for each task. For ECG and Engine, we construct traces using predefined templates populated with their original annotated step labels and short answers. For HAR, Sleep, and TSR, we generate reference traces using Qwen3.5-27B, following Messica et al. (2026). Full details and per-dataset statistics are provided in Appendix A.1 and Table 4.

Baselines. We compare Lapras against three classes of reasoning approaches. **(i) No-CoT** produces direct answers without generating reasoning traces. **(ii) Explicit CoT** generates step-by-step textual rationales before producing final answers, trained on reference reasoning traces. **(iii) Latent reasoning** includes two baselines: **iCoT** progressively removes CoT tokens during training so that the model learns to reason without explicit verbalization, and **COCONUT** progressively replaces textual steps with continuous hidden states through curriculum learning.

Backbones and Metrics. We evaluate all reasoning approaches across four TSLM backbones (ChatTS-7B, OpenTSLM-1B, SLIP-1B, and ITFormer-0.5B) on five reasoning tasks, where

Table 1: **Main Result.** Accuracy and Macro-F1 (%) of five finetuning methods on four TSLMs across five reasoning tasks. Our method is shaded in blue. **Bold** and underline denote the best and second-best per column within each model. Δ reports the change in average relative to No-CoT.

		ECG		Sleep		HAR		TSR		Engine		Avg.			
Model	Finetune Method	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	Δ	F1	Δ
ChatTS-7B	No-CoT	81.65	81.63	87.65	78.05	69.42	63.41	66.54	66.50	27.98	27.92	66.65		63.50	
	CoT	59.25	58.56	79.63	65.48	72.79	67.27	61.90	61.87	35.23	**35.49**	61.76	(-4.89)	57.73	(-5.77)
	Latent Reasoning:														
	iCoT	76.83	76.76	87.00	78.54	73.74	69.55	67.20	67.16	33.68	32.81	67.69	(+1.04)	64.96	(+1.46)
	COCONUT	84.45	84.41	86.67	77.31	73.52	68.36	65.58	65.56	**36.27**	35.24	69.30	(+2.65)	66.18	(+2.67)
	Lapras (Ours)	**85.23**	**85.17**	**88.19**	**79.49**	**77.43**	**73.53**	**69.32**	**69.32**	35.75	35.10	**71.18**	(+4.54)	**68.52**	(+5.02)
OpenTSLM-1B	No-CoT	71.07	70.90	77.57	65.25	71.31	66.22	33.29	33.32	28.50	11.09	56.35		49.36	
	CoT	65.47	65.33	72.81	58.02	67.61	61.71	52.69	53.13	27.98	26.48	57.31	(+0.96)	52.93	(+3.58)
	Latent Reasoning:														
	iCoT	72.94	72.78	77.57	65.22	69.98	65.40	51.10	51.08	26.94	24.20	59.71	(+3.36)	55.74	(+6.38)
	COCONUT	70.45	70.38	76.27	62.80	70.82	65.89	47.92	41.91	24.87	16.84	58.07	(+1.72)	51.56	(+2.21)
	Lapras (Ours)	**73.56**	**73.49**	**80.72**	**68.91**	**72.40**	**68.20**	**58.40**	**58.44**	**32.12**	**27.51**	**63.44**	(+7.09)	**59.31**	(+9.95)
SLIP-1B	No-CoT	71.38	71.35	77.57	64.36	73.74	69.20	50.90	50.78	37.82	**34.02**	62.28		57.94	
	CoT	72.01	71.87	70.53	55.88	70.49	66.71	62.16	62.17	34.20	31.54	61.88	(-0.40)	57.63	(-0.31)
	Latent Reasoning:														
	iCoT	81.03	80.96	77.36	62.76	72.70	68.71	61.58	61.58	22.28	13.26	62.99	(+0.71)	57.45	(-0.49)
	COCONUT	79.32	79.24	77.25	62.79	72.54	68.76	50.27	50.27	28.50	11.09	61.58	(-0.71)	54.43	(-3.51)
	Lapras (Ours)	**82.74**	**82.64**	**78.12**	**66.31**	**74.84**	**70.75**	**67.32**	**67.30**	**38.86**	26.49	**68.38**	(+6.09)	**62.70**	(+4.76)
ITFormer-0.5B	No-CoT	52.57	48.43	62.41	35.35	67.28	**59.23**	48.90	47.22	35.23	32.19	53.28		**44.48**	
	CoT	50.70	**50.27**	53.09	29.81	64.47	56.06	48.90	48.89	**35.75**	**35.73**	50.58	(-2.70)	44.15	(-0.33)
	Latent Reasoning:														
	iCoT	50.86	37.75	45.40	12.49	66.40	59.09	**55.98**	**55.98**	26.94	26.17	49.12	(-4.16)	38.30	(-6.19)
	COCONUT	51.48	34.55	45.50	12.51	**67.54**	55.92	47.39	42.41	31.61	31.03	48.70	(-4.57)	35.28	(-9.20)
	Lapras (Ours)	**53.81**	49.29	**63.06**	**36.95**	67.32	59.09	49.51	48.34	**35.75**	26.46	**53.89**	(+0.61)	44.03	(-0.46)

OpenTSLM-1B refers to its OpenTSLM-Flamingo variant (see Appendix A.3 for the OpenTSLM-SoftPrompt variant). Backbone details are provided in Appendix A.2. For evaluation, we parse the predicted class label from each model's generated output and compare it with the ground-truth label. We report accuracy and macro-F1 for each task.

5 Results and Discussion

5.1 Main Results

To evaluate the effectiveness of Lapras, we compare it with existing reasoning approaches on five sensor question answering datasets, where each task is formulated as multiple-choice classification over sensor time series. Table 1 reports accuracy and F1 for the five settings on four TSLM backbones ranging from 0.5B to 7B parameters. Generally, latent reasoning approaches improve performance compared with explicit CoT. In particular, Lapras achieves the highest average F1 on three of the four backbones. Compared with explicit CoT, Lapras improves F1 by 5.53% on average across the 20 backbone-dataset configurations, with the largest gain in average F1 of 10.79% on ChatTS-7B. We also note that Lapras achieves higher average F1 than both iCoT and COCONUT on every backbone. Its average F1 advantage over COCONUT is 6.78% across the four backbones, demonstrating the effectiveness of self-distillation compared to unconstrained continuous thought.

In addition, CoT underperforms No-CoT in average F1 on three of the four backbones, with an average drop of 2.14% on these backbones. This confirms the verbalization bottleneck identified in Section 3.1: forcing the model to describe fine-grained temporal patterns in text degrades the final answer F1. We further note that for ITFormer-0.5B, all reasoning methods including Lapras underperform No-CoT in average F1, suggesting that a 0.5B-parameter model cannot effectively learn from CoT supervision, consistent with findings by Li et al. (2025b). Taken together, Lapras consistently outperforms explicit CoT and existing latent reasoning methods across backbones and tasks, establishing it as a practical post-training reasoning paradigm for TSLMs.

5.2 Qualitative Analysis

Lapras's continuous thoughts can be decoded into readable reasoning traces. An advantage of explicit CoT reasoning is that it expresses intermediate steps in natural language, allowing users to inspect the model's reasoning trace. For example, predictions from models deployed in high-

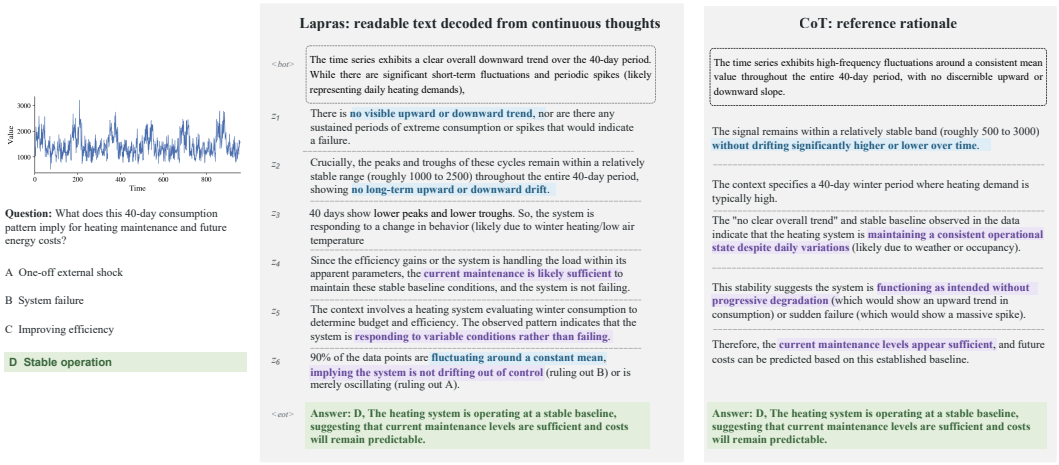


Figure 4: **Case study of decoded continuous thoughts of Lapras.** Lapras's continuous thoughts yield readable, task-relevant text whose content can be compared with the reference rationale. The decoded steps progress from a **signal description** of the series, through **inferences** built on it, to the **final answer**. Highlight colors match those in the figure.

stakes settings such as clinical monitoring need human verification before being acted upon. In latent reasoning models, however, these steps are represented as continuous hidden states and are not directly readable. We therefore investigate whether Lapras retains the ability to provide human-readable reasoning traces despite performing its intermediate computation in latent space. Benefiting from the knowledge distillation of Lapras, which directly supervises the continuous thoughts, we show that the continuous thoughts can be decoded into human-readable reasoning traces through standard LM decoding with the shared LM head. Specifically, we greedily decode each position $\langle \text{bot} \rangle, z_1, \ldots, z_6, \langle \text{eot} \rangle$ with the LM head. Figure 4 shows that the decoded steps contain global series descriptions, local segment descriptions, and inferences that lead to the answer, consistent with the reference reasoning trace. This suggests that readable traces can be retrieved post hoc from continuous thoughts, providing a way to audit the model's latent reasoning. Appendix A.7 reports additional cases.

Lapras mitigates the verbalization bottleneck by delaying answer commitment. We hypothesize that the verbalization bottleneck forces premature answer commitment at the first reasoning step, indicated by immediately assigning high probability to a single answer that remains unchanged with additional reasoning steps. Hence, mitigating this bottleneck should delay commitment, keeping probabilities balanced across multiple plausible answers early on before converging on the final answer. To evaluate this hypothesis, we track top-1 (most likely) and top-2 (two most likely) answer confidence across reasoning steps for Lapras and CoT, following Wang et al. (2025c). As shown in Figure 5, Lapras's top-1 probability stays well below the combined top-2 probability mass, indicating that the model considers multiple answers before gradually shifting its probability toward the correct choice. In contrast, CoT commits immediately at the first reasoning step, assigning near-total probability mass to a single answer. Because subsequent steps cannot recover from early descriptive errors, CoT depends entirely on the accuracy of its initial verbalization. This supports our preliminary findings (Section 3.1) and the logit-lens analysis in Appendix A.5.

Lapras tends to allocate more attention to time series representations during reasoning. To investigate whether avoiding intermediate verbalization encourages greater attention to the continuous temporal representations, we compare the attention patterns of Lapras and explicit CoT on a representative TSR example using ChatTS-7B (Appendix A.6). Figure 8 and Table 8 show that, in this example, Lapras assigns a larger share of attention to the time series at all three inspected layers. The difference is largest at the first layer, where Lapras allocates 17.35% of its attention to time series tokens, compared with 3.39% for CoT. These observations are consistent with Lapras drawing more directly on the original temporal representations, while CoT may neglect temporal patterns during verbalization.

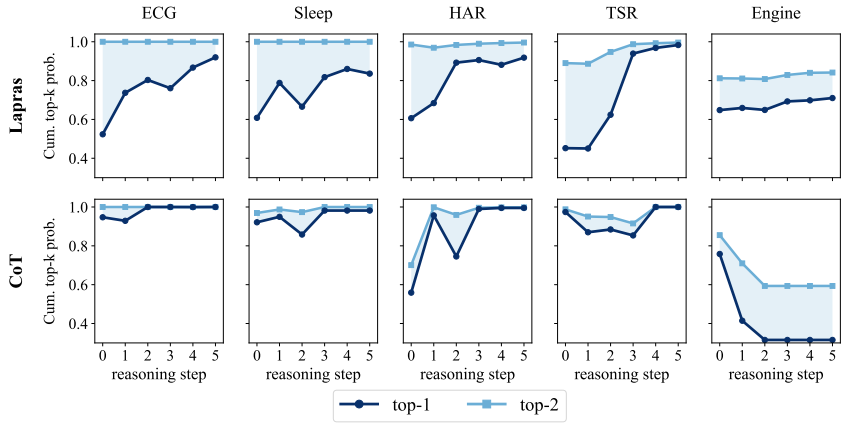


Figure 5: **Answer probability across reasoning steps** on the SLIP-1B backbone for Lapras (top) and CoT (bottom), shown for one representative example per task. Curves show the cumulative probability of **Answer in top-1** and **Answer in top-2**. Lapras keeps a second candidate alive well into the chain and commits near the end, whereas CoT tends to commit from the first step.

5.3 Quantitative Analysis

Inference efficiency. In CoT, the model autoregressively decodes each reasoning step as text tokens, so inference cost scales linearly with the number of generated tokens. A typical CoT trace can span hundreds of tokens, whereas Lapras compresses the entire reasoning process into only $K{=}6$ continuous thoughts, generating text tokens only for the final answer. This results in 6.9 to 23.9× fewer generated tokens and 5.9 to 15.1× faster inference than CoT (Figure 6), while adding less than 8 ms overhead compared to No-CoT. This makes latent reasoning practical for latency-sensitive deployment settings such as real-time health monitoring, where predictions must arrive within strict time constraints.

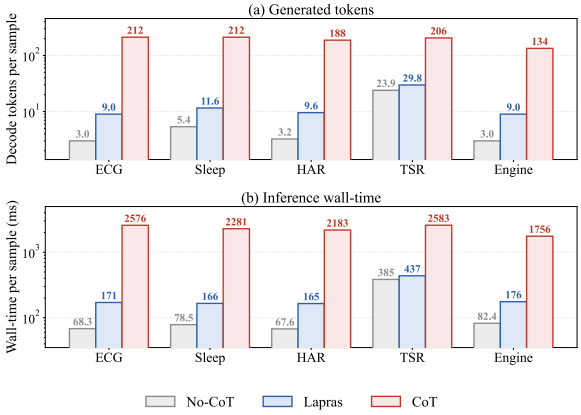


Figure 6: **Inference efficiency of the three reasoning methods.** Generated tokens and wall-clock time per sample from SLIP-1B with one NVIDIA A6000.

Number of continuous thoughts. Lapras emits a fixed number of continuous thoughts regardless of the task. Understanding which tasks benefit from these thoughts helps identify where latent reasoning provides value. We test this by taking the model finetuned with $K{=}6$ and varying the number of generated continuous thoughts at inference. As Table 2 shows, using 6 continuous thoughts raises average accuracy from 64.82% to 68.38% compared to using a single continuous thought. Increasing to $K{=}8$ does not improve further, with average accuracy slightly declining to 68.07%, indicating that additional thoughts beyond the training configuration yield no benefit. The largest gains are on TSR (+12.46%) and Engine (+4.66%). ECG, Sleep, and HAR change by at most 1.09%. We attribute this to these tasks depending on local pattern recognition rather than on multi-step reasoning. This suggests that the benefit of continuous thoughts scales with the amount of multi-step inference a task requires after the time series has been encoded.

5.4 Ablation Studies

To understand which design choices contribute to Lapras's performance, we ablate three components on SLIP-1B across all five tasks (Table 3): **latent projection**, which maps continuous thoughts back into the input embedding space for the next reasoning step; **reasoning distillation**, which enables

Table 2: **Number of continuous thoughts.** Answer accuracy (%) with SLIP-1B, varying only the number of continuous thoughts K. Default K=6 in blue. Δ reports the change in average relative to K=1.

	ECG	Sleep	HAR	TSR	Engine	Avg.	Δ
K=1	81.65	78.22	75.18	54.86	34.20	64.82	
K=3	81.65	77.36	75.05	64.44	33.16	66.33	(+1.51)
K=6	82.74	78.12	74.84	67.32	38.86	68.38	(+3.56)
K=8	82.43	77.36	74.85	67.39	38.34	68.07	(+3.25)

the student to acquire the teacher's reasoning ability without generating explicit CoT; and **masking**, which regularizes training.

Removing the latent projection π causes the largest drop, reducing average accuracy from 68.38% to 62.23% (-6.15%). This shows that learning a transformation between the hidden states produced by one reasoning step and the inputs consumed by the next is critical for effective latent reasoning. Removing distillation while retaining the latent reasoning steps reduces average accuracy to 64.34% (-4.04%). Without distillation, the continuous thoughts receive no supervision and the model is trained with only the final-answer cross-entropy loss, making it functionally equivalent to a No-CoT model with additional forward passes. The drop confirms that supervising the continuous thoughts through distillation is necessary for the model to learn effective latent representations. Finally, removing masking reduces average accuracy to 66.58% (-1.80%), suggesting it prevents the model from bypassing the latent reasoning steps.

Table 3: **Lapras ablation experiments** with the SLIP-1B backbone on the five reasoning benchmarks. We report answer accuracy (%). Default settings are marked in blue.

Method	ECG	Sleep	HAR	TSR	Engine	Avg.	Δ
Lapras (SLIP-1B) (Ours)	82.74	78.12	74.84	67.32	38.86	68.38	
w/o projection π	80.87	77.68	73.60	50.00	29.02	62.23	(-6.15)
w/o reasoning distillation	78.54	77.25	74.45	56.23	35.23	64.34	(-4.04)
w/o mask	79.41	78.44	72.18	69.17	33.68	66.58	(-1.80)

6 Conclusion

Limitations and future work. Our work has several limitations. First, Lapras still relies on explicit CoT annotations for training as they are required by the teacher, restricting its use to settings where such data can be obtained or synthetically generated. Second, the number of latent reasoning steps is fixed at $K = 6$ regardless of task or input complexity. Learning to adaptively allocate computation per input, for instance through reinforcement learning over the number of continuous thoughts, is a natural extension. Third, on our smallest backbone (ITFormer-0.5B), all reasoning methods, including Lapras, underperform No-CoT in average F1, indicating that multi-step reasoning may require a minimum model capacity to be effective. Understanding how this capacity threshold interacts with model architecture and task complexity remains an open question.

Conclusion and broader impact. Our experiments show that finetuning TSLMs with explicit CoT reasoning does not consistently improve average F1 over finetuning with direct answers alone. We hypothesize that this is due to the verbalization bottleneck. Specifically, when a TSLM must verbalize a continuous time series before reasoning over it, subsequent reasoning steps are conditioned on incomplete or inaccurate textual descriptions rather than on the original signal, which degrades downstream performance. Lapras addresses this by performing reasoning in the model's joint time series–language space instead of in text. Across four backbones from 0.5B to 7B parameters, we find that Lapras improves average F1 over CoT finetuning by up to 10.79% (ChatTS-7B), while reducing the number of generated tokens by up to 95.8%. Qualitative analyses further show that the continuous thoughts produced by Lapras can be decoded into readable textual traces through standard language decoding. More broadly, as TSLMs are increasingly applied to safety-critical domains such as clinical monitoring and industrial diagnostics, Lapras offers a post-training paradigm that improves both reasoning accuracy and inference efficiency without sacrificing interpretability.

AI USE STATEMENT

In this work, we used generative AI tools to aid and polish writing, for retrieval and discovery (e.g., finding related work), and for generating synthetic datasets. We have reviewed all AI-assisted work. Specifically, all AI-polished text was manually reviewed and revised by the authors to ensure accuracy and faithful representation of our contributions. Retrieved references were verified against original sources, and all synthetic datasets were validated for correctness before use in experiments. We did not use generative AI tools for research ideation or execution, to draft sections of the paper, or to prove mathematical claims. The remaining required disclosure categories are not applicable to this work. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

To support reproducibility, we provide complete implementation details in the appendix, including model architecture specifications, hyperparameter configurations, training procedures, and dataset construction steps. All experimental settings required to reproduce the reported results are documented therein. We will release our code and datasets upon acceptance.

REFERENCES

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katie Millican, Malcolm Reynolds, Roman Ring, Eliza Rutherford, Serkan Cabi, Tengda Han, Zhitao Gong, Sina Samangooei, Marianne Monteiro, Jacob Menick, Sebastian Borgeaud, Andrew Brock, Aida Nematzadeh, Sahand Sharifzadeh, Mikolaj Binkowski, Ricardo Barreira, Oriol Vinyals, Andrew Zisserman, and Karen Simonyan. Flamingo: a visual language model for few-shot learning, 2022. URL https://arxiv.org/abs/2204.14198.

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, 2024. URL https://arxiv.org/abs/2403.07815.

Mohammad Asadi, Jack W. O'Sullivan, Fang Cao, Tahoura Nedaee, Kamyar Rajabalifardi, Fei-Fei Li, Ehsan Adeli, and Euan Ashley. Mirage: The illusion of visual understanding, 2026. URL https://arxiv.org/abs/2603.21687.

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers, 2021. URL https://arxiv.org/abs/2104.14294.

Liang Chen, Haozhe Zhao, Tianyu Liu, Shuai Bai, Junyang Lin, Chang Zhou, and Baobao Chang. An image is worth 1/2 tokens after layer 2: Plug-and-play inference acceleration for large vision-language models, 2024. URL https://arxiv.org/abs/2403.06764.

Yuliang Chen, Arvind Pillai, Yu Yvonne Wu, Tess Z. Griffin, Lisa Marsch, Michael V. Heinz, Nicholas C. Jacobson, and Andrew Campbell. Learning transferable sensor models via language-informed pretraining, 2026. URL https://arxiv.org/abs/2603.11950.

Abhimanyu Das, Weihao Kong, Rajat Sen, and Yichen Zhou. A decoder-only foundation model for time-series forecasting, 2024. URL https://arxiv.org/abs/2310.10688.

Jingcheng Deng, Liang Pang, Zihao Wei, Shicheng Xu, Zenghao Duan, Kun Xu, Yang Song, Huawei Shen, and Xueqi Cheng. Llm latent reasoning as chain of superposition, 2026. URL https://arxiv.org/abs/2510.15522.

Jun Gao, Yongqi Li, Ziqiang Cao, and Wenjie Li. Interleaved-modal chain-of-thought, 2025. URL https://arxiv.org/abs/2411.19488.

Mononito Goswami, Konrad Szafer, Arjun Choudhry, Yifu Cai, Shuo Li, and Artur Dubrawski. Moment: A family of open time-series foundation models, 2024. URL https://arxiv.org/abs/2402.03885.

Nate Gruver, Marc Finzi, Shikai Qiu, and Andrew Gordon Wilson. Large language models are zero-shot time series forecasters, 2024. URL https://arxiv.org/abs/2310.07820.

Tong Guan, Zijie Meng, Dianqi Li, Shiyu Wang, Chao-Han Huck Yang, Qingsong Wen, Zuozhu Liu, Sabato Marco Siniscalchi, Ming Jin, and Shirui Pan. Timeomni-1: Incentivizing complex reasoning with time series in large language models, 2026. URL https://arxiv.org/abs/2509.24803.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space, 2025. URL https://arxiv.org/abs/2412.06769.

Yubin Kim, Xuhai Xu, Daniel McDuff, Cynthia Breazeal, and Hae Won Park. Health-llm: Large language models for health prediction via wearable sensor data, 2024. URL https://arxiv.org/abs/2401.06866.

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, 2026. URL https://arxiv.org/abs/2510.02410.

Tamera Lanham, Anna Chen, Ansh Radhakrishnan, Benoit Steiner, Carson Denison, Danny Hernandez, Dustin Li, Esin Durmus, Evan Hubinger, Jackson Kernion, Kamilė Lukošiūtė, Karina Nguyen, Newton Cheng, Nicholas Joseph, Nicholas Schiefer, Oliver Rausch, Robin Larson, Sam McCandlish, Sandipan Kundu, Saurav Kadavath, Shannon Yang, Thomas Henighan, Timothy Maxwell, Timothy Telleen-Lawton, Tristan Hume, Zac Hatfield-Dodds, Jared Kaplan, Jan Brauner, Samuel R. Bowman, and Ethan Perez. Measuring faithfulness in chain-of-thought reasoning, 2023. URL https://arxiv.org/abs/2307.13702.

Bangzheng Li, Ximeng Sun, Jiang Liu, Ze Wang, Jialian Wu, Xiaodong Yu, Hao Chen, Emad Barsoum, Muhao Chen, and Zicheng Liu. Latent visual reasoning, 2025a. URL https://arxiv.org/abs/2509.24251.

Yuetai Li, Xiang Yue, Zhangchen Xu, Fengqing Jiang, Luyao Niu, Bill Yuchen Lin, Bhaskar Ramasubramanian, and Radha Poovendran. Small models struggle to learn from strong reasoners, 2025b. URL https://arxiv.org/abs/2502.12143.

Xin Liu, Daniel McDuff, Geza Kovacs, Isaac Galatzer-Levy, Jacob Sunshine, Jiening Zhan, Ming-Zher Poh, Shun Liao, Paolo Di Achille, and Shwetak Patel. Large language models are few-shot health learners, 2023. URL https://arxiv.org/abs/2305.15525.

Yunfei Luo, Yuliang Chen, Asif Salekin, and Tauhidur Rahman. Toward foundation model for multivariate wearable sensing of physiological signals, 2025. URL https://arxiv.org/abs/2412.09758.

Mike A Merrill, Mingtian Tan, Vinayak Gupta, Thomas Hartvigsen, and Tim Althoff. Language models still struggle to zero-shot reason about time series. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *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/.

Shvat Messica, Jiawen Zhang, Kevin Li, Theodoros Tsiligkaridis, and Marinka Zitnik. Adaptive time series reasoning via segment selection, 2026. URL https://arxiv.org/abs/2602.18645.

Jungwoo Oh, Hyunseung Chung, Junhee Lee, Min-Gyu Kim, Hangyul Yoon, Ki Seong Lee, Youngchae Lee, Muhan Yeo, and Edward Choi. Ecg-reasoning-benchmark: A benchmark for evaluating clinical reasoning capabilities in ecg interpretation, 2026. URL https://arxiv.org/abs/2603.14326.

Zhenyi Shen, Hanqi Yan, Linhai Zhang, Zhanghao Hu, Yali Du, and Yulan He. Codi: Compressing chain-of-thought into continuous space via self-distillation, 2025. URL https://arxiv.org/abs/2502.21074.

Miles Turpin, Julian Michael, Ethan Perez, and Samuel R. Bowman. Language models don't always say what they think: Unfaithful explanations in chain-of-thought prompting, 2023. URL https://arxiv.org/abs/2305.04388.

Qixun Wang, Yang Shi, Yifei Wang, Yuanxing Zhang, Pengfei Wan, Kun Gai, Xianghua Ying, and Yisen Wang. Monet: Reasoning in latent visual space beyond images and language, 2025a. URL https://arxiv.org/abs/2511.21395.

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, 2025b. URL https://arxiv.org/abs/2506.20093.

Zezhong Wang, Xingshan Zeng, Weiwen Liu, Yufei Wang, Liangyou Li, Yasheng Wang, Lifeng Shang, Xin Jiang, Qun Liu, and Kam-Fai Wong. Chain-of-probe: Examining the necessity and accuracy of cot step-by-step, 2025c. URL https://arxiv.org/abs/2406.16144.

Qiong Wu, Xiangcong Yang, Yiyi Zhou, Chenxin Fang, Baiyang Song, Xiaoshuai Sun, and Rongrong Ji. Grounded chain-of-thought for multimodal large language models, 2025. URL https://arxiv.org/abs/2503.12799.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks, 2024. URL https://arxiv.org/abs/2309.17453.

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, April 2025. ISSN 2150-8097. doi: 10.14778/3742728.3742735. URL http://dx.doi.org/10.14778/3742728.3742735.

Maxwell A. Xu, Harish Haresamudram, Catherine W. Liu, Patrick Langer, Jathurshan Pradeepkumar, Wanting Mao, Sunita J. Ferns, Aradhana Verma, Jimeng Sun, Paul Schmiedmayer, Xin Liu, Daniel McDuff, Emily B. Fox, and James M. Rehg. How well do multimodal models reason on ecg signals?, 2026. URL https://arxiv.org/abs/2603.00312.

Muru Zhang, Ofir Press, William Merrill, Alisa Liu, and Noah A. Smith. How language model hallucinations can snowball, 2023. URL https://arxiv.org/abs/2305.13534.

Xiaoyu Zhu, Xinke Deng, Suresh Taddewadikar, Arnab Kumar Mondal, Zhongyu Jiang, Ian Fasel, and Joerg Liebelt. Beyond visual cot: Internalized visual thinking for proactive video reasoning, 2026. URL https://arxiv.org/abs/2608.15869.

A Appendix

A.1 Dataset Details

We construct a reasoning trace for every task, since Lapras is trained by distilling one. The five tasks divide by how that trace is obtained: ECG and Engine carry certified, stage-wise ground-truth labels, so their traces are assembled deterministically from those labels; HAR, Sleep, and TSR do not, so their traces are distilled from a teacher model. In all cases the trace is used only to shape training and is never emitted at inference. Table 4 reports the split sizes, channel counts, series lengths, and answer format of each task.

ECG. The original ECG-Reasoning benchmark is multi-stage: each reasoning step is annotated separately and posed as an independent question. We reformat it into a single binary diagnosis question (e.g., presence vs. absence of atrial fibrillation) by concatenating the per-stage gold answers into one trace that proceeds from criterion selection, to finding identification, to ECG grounding, and finally to the diagnostic decision, which serves as the answer.

Engine. Engine is an aero-engine sensor dataset. We keep only the subset of samples that carry a certified severity label corresponding to each sensor channel, since only these samples support a verifiable ground-truth trace. From these certified labels the rationale is generated deterministically as a multi-step trace: a per-component roll-call over the five rotating components, a severity read-off, and the maintenance implication that follows from the resulting (component, severity) pair.

HAR and Sleep. HAR and Sleep are a human activity recognition and a sleep stage classification task, respectively; both require reasoning over time series to answer questions about the underlying activity or sleep stage. Following Messica et al. (2026), we generate the supervised finetuning rationales through a distillation pipeline. We prompt Qwen3.5-27B to answer each question while conditioning it on rendered plots of the signal, the summary statistics, and the gold answer. The model is first asked to identify at most three contiguous segments that are essential for answering the question, and then to produce a step-by-step explanation that reaches the answer using only those segments and the statistics. For each segment, the rationale for selecting it is stated separately from the analysis of what the segment reveals, which keeps the segment selection decoupled from its interpretation. We transform the resulting responses into structured traces that interleave natural language reasoning with explicit segment-selection calls, then post-process them to check correctness and adherence to the template.

TSR. TSR is the counterfactual subset of the original TSR benchmark (Merrill et al., 2024), a multiple-choice reasoning task over mixed-domain real-world signals (e.g., residential solar generation or city temperature). Given a signal, the model must predict a real-world consequence or judgment whose correct answer flips with the signal's qualitative behavior. Each item pairs an original series with its counterfactual counterpart; the two share the same question and answer choices but differ in the time series and in the correct answer, so the answer can only be determined by reading the series. As the dataset does not provide a reasoning rationale, we generate one with a distillation pipeline following Messica et al. (2026), as for HAR and Sleep.

Table 4: Overview of datasets.

Dataset	Train	Test	Channels	Len	Format
ECG (Oh et al., 2026)	5,800	643	12	1000	Binary (Yes/No)
HAR (Langer et al., 2026)	68,542	8,222	3	128	Multiple choice
Sleep (Langer et al., 2026)	7,413	923	1	1500	Multiple choice
TSR (Merrill et al., 2024)	22,566	4,094	1	128–1020	Multiple choice
Engine (Wang et al., 2025b)	2,305	193	33	600	Multiple choice

A.2 Backbone Architectures

All four backbones follow a common three-stage template. A pretrained language model f_ϕ is extended to a second modality by an encoder g_ψ that maps the series X_s to features Z_s, a connector

Table 5: **The four backbones along three design axes.** $|H_s|$ is the number of sensor tokens handed to the language model; C is the channel count, L the series length, and P the patch size of Table 7. *Concat* places H_s in the token sequence and reads it with self-attention; *cross-attn* keeps it outside and injects it through dedicated layers.

Backbone	LM	Sensor encoder	Connector	$\lvert H_s\rvert$	Fusion (depth)
ChatTS-7B	7B	per-patch MLP	none (native space)	$C\lceil L/P\rceil$	concat (all layers)
ITFormer-0.5B	0.5B	PatchTST	question-cond. resampler	25	concat (all layers)
SLIP-1B	1B	language-pretrained	attention pooler	64	cross-attn (last 4)
OpenTSLM-1B	1B	PatchTST	Perceiver resampler	64	cross-attn (every layer)

h_ω that maps these features into the language embedding space as $H_s = h_\omega(Z_s)$, and a fusion mechanism that determines where inside f_ϕ the sensor tokens are read. All four patch the input. A length-L, C-channel recording is divided into windows of P timesteps, analogous to image patches. The backbones differ along three axes: the patch encoding, the sensor token count, and the fusion mechanism. Table 5 summarizes the four configurations.

Encoder. ChatTS (Xie et al., 2025) min-max normalizes each series, writes the scale and offset into the prompt, and applies a five-layer MLP to each patch in isolation, so cross-time interaction occurs only inside the language model. ITFormer (Wang et al., 2025b) uses a PatchTST encoder, pretrained separately and frozen, that attends across the patches of a channel, so its features carry local temporal context. OpenTSLM (Langer et al., 2026) embeds each patch with a jointly trained 1D convolution, one channel at a time, and defers temporal mixing to the connector. SLIP (Chen et al., 2026) uses an encoder pretrained on large-scale sensor caption pairs with contrastive and captioning objectives, with a FlexMLP that adapts patch granularity to the sampling rate, so its features are aligned to language before QA finetuning.

Connector. ChatTS uses no connector. The MLP emits vectors in the language embedding space, and the patches of each series are inserted at the position where the series is named in the prompt. Every patch of every channel becomes one token, so $|H_s| = C\lceil L/P\rceil$ grows with the channel count and the series length. Table 7 therefore uses a larger patch size on the 12-channel ECG and 33-channel Engine tasks, where $P = 8$ yields over a thousand sensor tokens and exceeds the memory available at the 7B scale. The other three backbones use a resampler, a fixed set of learned queries that cross-attend to the encoder features and return one token each. The token count is fixed by the architecture and independent of the series length, 25 for ITFormer and 64 for SLIP and OpenTSLM. ITFormer prepends 25 learnable instruct tokens to the question, refines them with self-attention, and uses them as queries that pool the encoder features across channels and then across timesteps, so the returned tokens depend on the instruction. SLIP and OpenTSLM use question-agnostic resamplers, a Perceiver resampler in OpenTSLM and caption queries pretrained with the encoder in SLIP. The 64 tokens are computed from the series alone, and the question interacts with them only when the text tokens cross-attend to H_s inside f_ϕ.

Fusion. ChatTS and ITFormer concatenate the sensor tokens into the prompt, $H = f_\phi([H_q; H_s])$. The signal is read by self-attention at every layer, no parameters are added to the language model, and the two modalities share one attention budget. SLIP and OpenTSLM keep H_s outside the sequence and inject it through gated cross-attention layers interleaved into f_ϕ, $H = f_\phi(H_q \mid H_s)$, following Flamingo (Alayrac et al., 2022). The sensor pathway has separate parameters and a separate attention budget, a tanh gate initialized at zero preserves the pretrained text behavior at initialization, and memory is constant in the series length. OpenTSLM injects at every layer. SLIP injects at the final four layers, so lower layers process the question as text alone.

Coverage. The four backbones span all three axes and language models from 0.5B to 7B parameters. Lapras modifies none of these components and changes only the tokens the language model emits between the prompt and the answer, so evaluating on all four separates its effect from any particular integration of the signal. The attention analysis of Appendix A.6 requires a concatenation backbone, because λ_{ts} is defined as a share of a single self-attention row and is measurable only when the sensor tokens are in the sequence; we use ChatTS-7B as a representative one. The latent

loop of Equation 1, by contrast, applies to all four backbones. Each continuous thought z_k re-enters f_ϕ as an input embedding and reads the signal through the fusion pathway of the backbone, whereas an explicit chain conditions increasingly on its own generated text.

A.3 Architecture Variant Discussion

We briefly explore whether the effect of the finetuning method changes across architecture families. OpenTSLM has released two controlled variants, OpenTSLM-Flamingo and OpenTSLM-SoftPrompt, which share the same 1B language model but differ in how the sensor tokens interact with it (Appendix A.2). This pair gives us an opportunity for a controlled experiment that isolates the fusion pathway while keeping all other factors fixed.

As shown in Table 6, OpenTSLM-SoftPrompt outperforms OpenTSLM-Flamingo in average accuracy under every finetuning method, though which variant leads on any given task varies without a consistent pattern. Lapras improves accuracy on both variants and narrows the gap between them from 3.57% under No-CoT to 0.67%, suggesting that latent reasoning reduces the model's sensitivity to how sensor tokens are fused with the language backbone. Explicit chain-of-thought, by contrast, does not produce a clear benefit on either variant.

Overall, the gains from latent reasoning are consistent across both fusion pathways, and the choice between them matters less under Lapras than under No-CoT. However, whether this behavior holds for larger language backbones (e.g., 7B) remains open for future work.

Table 6: **Effect of the fusion pathway.** Accuracy and Macro-F1 (%) on the two OpenTSLM variants, which share a 1B language model and differ only in fusion. Ours in blue ; **bold** and underline are best and second-best per column within a variant; Δ is the change in average over No-CoT.

		ECG		Sleep		HAR		TSR		Engine		Avg.			
Model	Finetune Method	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	Δ	F1	Δ
OpenTSLM-SoftPrompt	No-CoT	61.59	60.98	84.29	73.93	73.46	69.85	49.15	48.65	31.09	30.64	59.92		56.81	
	CoT	52.10	51.94	83.42	72.87	71.14	67.00	53.03	53.02	32.12	32.11	58.36	(-1.55)	55.39	(-1.42)
	iCoT	63.76	63.62	84.07	72.10	72.55	69.34	54.86	54.85	26.42	18.03	60.33	(+0.41)	55.59	(-1.22)
	COCONUT	**65.16**	64.48	83.42	71.25	74.18	**71.18**	56.74	56.50	34.72	33.83	62.84	(+2.93)	59.45	(+2.64)
	Lapras (Ours)	**65.16**	**64.80**	**85.81**	**75.62**	**74.25**	70.94	**58.04**	**58.00**	**37.31**	**36.08**	**64.11**	(+4.20)	**61.09**	(+4.28)
OpenTSLM-Flamingo	No-CoT	71.07	70.90	77.57	65.25	71.31	66.22	33.29	33.32	28.50	11.09	56.35		49.36	
	CoT	65.47	65.33	72.81	58.02	67.61	61.71	52.69	53.13	27.98	26.48	57.31	(+0.96)	52.93	(+3.58)
	iCoT	72.94	72.78	77.57	65.22	69.98	65.40	51.10	51.08	26.94	24.20	59.71	(+3.36)	55.74	(+6.38)
	COCONUT	70.45	70.38	76.27	62.80	70.82	65.89	47.92	41.91	24.87	16.84	58.07	(+1.72)	51.56	(+2.21)
	Lapras (Ours)	**73.56**	**73.49**	**80.72**	**68.91**	**72.40**	**68.20**	**58.40**	**58.44**	**32.12**	**27.51**	**63.44**	(+7.09)	**59.31**	(+9.95)

A.4 Implementation Details

Training configuration. All runs use LLaMA-Factory[1] with the AdamW optimizer, a cosine learning-rate schedule with warmup ratio 0.02, bf16 precision, DeepSpeed ZeRO-2, gradient clipping at 1.0, and an effective batch size of 128. In all settings, the TSLM is fully finetuned. The Lapras runs use the default configuration with the latent projector enabled, $K = 6$ continuous thoughts, and a mask ratio of 0.3 on patchified sensor streams during training (Section 3.2). Table 7 reports the settings that vary across backbones and tasks.

We start from the recommended hyperparameters of each backbone (Xie et al., 2025; Wang et al., 2025b; Langer et al., 2026; Chen et al., 2026) and make only the adjustments needed for the training loss to flatten. Within each backbone, the learning rate and weight decay are held fixed across all five finetuning methods. The only method-specific setting is the distillation weight λ of Lapras. We default to $\lambda = 10$ following CODI (Shen et al., 2025) and adjust it within $\{1, 10, 20\}$ so that the weighted distillation loss stays on the same order of magnitude as the student and teacher cross-entropy losses.

Patch sizes follow each backbone's default. OpenTSLM-Flamingo, OpenTSLM-SoftPrompt, ChatTS, and ITFormer use fixed patch sizes of 4, 4, 8, and 60, respectively. We slightly adjust them for computational efficiency. For ChatTS, we increase the patch size on ECG and Engine to 32 and 50 to avoid memory overflow at the 7B scale. For OpenTSLM-SoftPrompt, we increase the

[1] https://github.com/hiyouga/LLaMA-Factory

Table 7: **Per-task hyperparameters** for the four backbones and the OpenTSLM-SoftPrompt variant. One row covers all five finetuning methods, which share every setting except the distillation weight λ of Lapras.

Backbone	Dataset	Patch Size	Epochs	lr	wd	λ
ChatTS-7B	ECG	32	20	1e-5	1e-2	10
	Sleep	8	20	1e-5	1e-2	10
	HAR	8	4	1e-5	1e-2	10
	TSR	8	4	1e-5	1e-2	1
	Engine	50	5	1e-5	1e-2	20
SLIP-1B	ECG	32	20	5e-5	1e-2	10
	Sleep	64	20	1e-4	1e-2	10
	HAR	4	4	1e-4	1e-2	10
	TSR	16/32	4	1e-4	1e-2	10
	Engine	32	5	1e-5	1e-2	20
OpenTSLM-Flamingo	ECG	4	20	1e-4	1e-2	10
	Sleep	4	20	1e-4	1e-2	1
	HAR	4	4	1e-4	1e-2	1
	TSR	4	4	1e-5	1e-2	10
	Engine	4	5	1e-5	1e-2	10
OpenTSLM-SoftPrompt	ECG	100	20	1e-4	1e-2	10
	Sleep	4	20	1e-4	1e-2	1
	HAR	4	4	1e-4	1e-2	1
	TSR	4	4	1e-5	1e-2	10
	Engine	50	5	1e-5	1e-2	10
ITFormer-0.5B	ECG	60	20	1e-5	1e-2	10
	Sleep	60	20	1e-5	1e-2	10
	HAR	60	4	1e-5	1e-2	10
	TSR	60	4	1e-5	1e-2	10
	Engine	60	5	1e-5	1e-2	10

patch size on the same two datasets to 100 and 50 so that its effective batch size matches that of OpenTSLM-Flamingo. SLIP uses an adaptive patch size per dataset, following its own recommendation.

For each dataset, the epoch count is shared across all backbones and all five finetuning methods, and we report the final checkpoint with no checkpoint selection, so no test-set information enters the schedule. Evaluation uses greedy decoding with at most 512 new tokens for every method. Each run uses 4×H200.

A.5 Per-Token Predictive Entropy via the Logit Lens

Setup. To characterize *what a continuous thought encodes*, we decode every intermediate state through the model's own output head, following the logit-lens analysis of latent reasoning (Deng et al., 2026). A language model maps any hidden state h_t to a next-token distribution with the same unembedding it uses at the output layer, $p_t = \mathrm{softmax}(W_U h_t/\tau)$, where $W_U \in \mathbb{R}^{V\times d}$ is the (tied) language-model head, V the vocabulary size, and $\tau > 0$ the temperature. We quantify how concentrated this distribution is with the Shannon entropy in bits, $H(p_t) = -\sum_{v=1}^{V} p_t(v)\log_2 p_t(v)$. A low $H(p_t)$ indicates that the state places nearly all probability mass on a single token, whereas a high $H(p_t)$ indicates that the state distributes mass across multiple candidate continuations and has not yet committed to one of them.

The readout is identical in both decoding modes, and only the input to W_U differs. In explicit decoding, h_t is computed after a discrete token has been embedded and fed back, so the $\arg\max$ of p_t is the emitted token and the lens recovers a prediction the model has already committed to. In latent decoding, the model emits no token during reasoning. Each continuous thought z_k is the last-layer hidden state, which is mapped through $\pi(\cdot)$ and fed back as the input embedding for the next step. The distribution $p_{z_k} = \mathrm{softmax}(W_U z_k/\tau)$ is therefore a *diagnostic* readout that indicates which vocabulary tokens the thought would favor if it were decoded at that step. Because the thought is never discretized, a single latent step can retain probability mass on several candidate tokens rather than on one. We summarize this spread by the *effective number of candidate tokens* per thought, $N_{\text{eff}} = 2^{\bar{H}}$, where $\bar{H}$ is the mean of $H(p_{z_k})$ over the K thoughts. N_{eff} equals the number of equiprobable tokens whose entropy matches $\bar{H}$, and $N_{\text{eff}} = 1$ corresponds to a deterministic prediction. For each example, we record $H(p_t)$ and the sorted top-k probabilities of p_t at every position, and we define the *commitment point* as the first answer token whose top-1 probability is at least 0.95.

Algorithm 1 Logit-lens entropy readout for explicit and latent decoding.

```
import torch, torch.nn.functional as F

# W_U: LM head, tau: temperature, h: last-layer state at last position
def logit_lens(h, tau=1.0):
   return F.softmax(W_U(h).float() / tau, dim=-1) # (V,)

def entropy_bits(p):
   return -(p * p.clamp_min(1e-12).log2()).sum()

def record(h, tag): # log entropy, top-k, phase
   p = logit_lens(h)
   H_seq.append(entropy_bits(p)); heat.append(p.topk(30).values); phase.append(tag)
   return p

def decode_answer(h): # greedy token decoding
   for _ in range(max_new_tokens):
      tok = record(h, "answer").argmax()
      if tok == eos: break
      h = step(embed(tok))[-1]

# ---- CoT ----
H_seq, heat, phase = [], [], []
decode_answer(encode(prompt)[-1])

# ---- latent reasoning ----
H_seq, heat, phase = [], [], []
h = encode(prompt + [bot])[-1]
for k in range(K):
   record(h, "latent") # h is thought z_k
   h = step(pi(h))[-1]
h = step(embed(eot))[-1] # end thinking
decode_answer(h)

# effective number of candidate tokens per continuous thought
H_lat = torch.stack([H for H, ph in zip(H_seq, phase) if ph == "latent"])
N_eff = 2 ** H_lat.mean()

# commitment point: first answer token with top-1 prob >= 0.95
commit = next((i for i, ph in enumerate(phase)
           if ph == "answer" and heat[i][0] >= 0.95), None)
```

Result. Figure 7 compares CoT and latent reasoning on the same example. Under CoT (top), the readout has low entropy at nearly all of the 140 steps. Each emitted token is already committed when it is produced, so the model reaches the answer through a long sequence of near-deterministic steps. Under latent reasoning (bottom), the computation is concentrated in 6 continuous thoughts whose readouts spread probability over many vocabulary tokens ($N_{\text{eff}} \approx 22$), which indicates that each thought retains mass on several candidate continuations rather than committing to a single token. Entropy decreases after the first thought, and the distribution reaches the commitment point (top-1 ≥ 0.95) only at the first answer token. The continuous thoughts therefore do not reproduce a compressed version of the CoT trajectory, and instead follow an entropy profile that differs from explicit decoding.

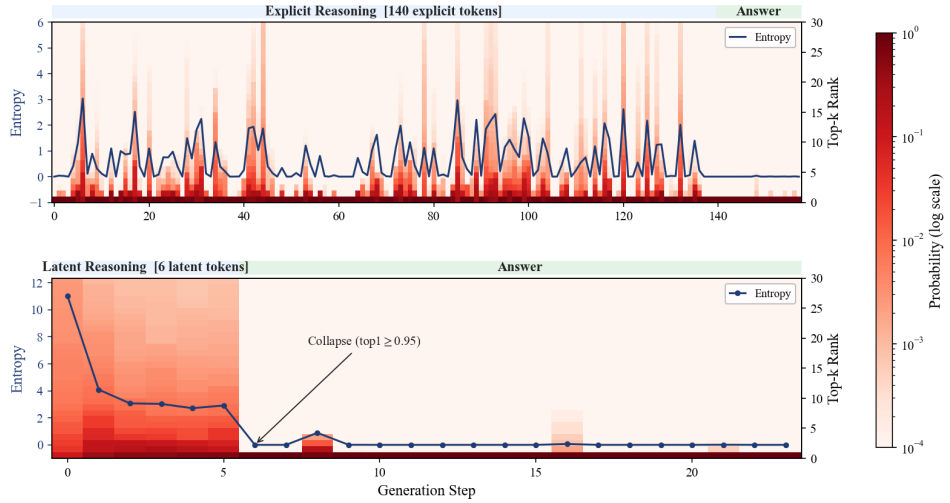


Figure 7: **Per-step logit-lens readout on one TSR example.** At each step, we plot the next-token distribution read through the logit lens, with tokens ranked by probability and colored on a log scale, together with its entropy (blue). **Top:** under CoT the distribution is sharply peaked at nearly all of the 140 steps. **Bottom:** under latent reasoning the 6 continuous thoughts spread probability over many tokens ($N_{\text{eff}} \approx 22$), entropy decreases after the first thought, and the distribution reaches the commitment point (top-1 ≥ 0.95) only at the first answer token.

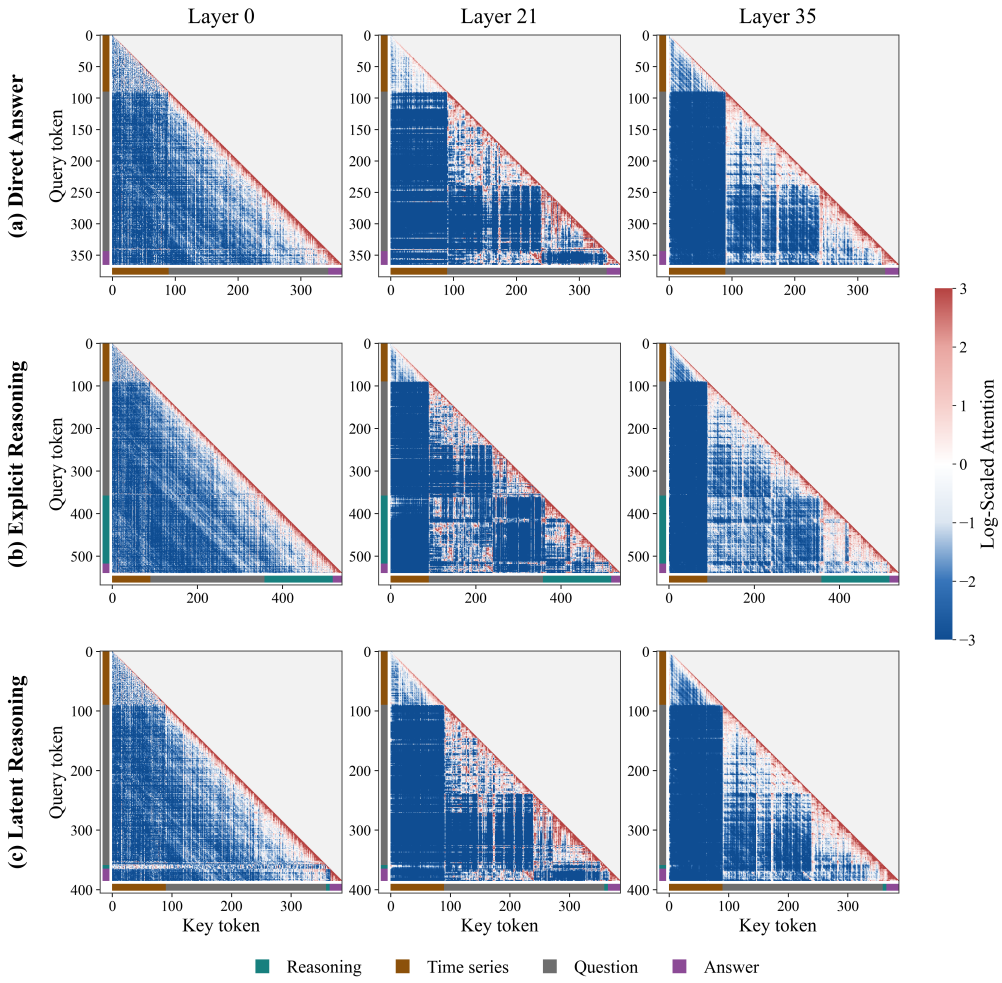


Figure 8: **Attention maps** of ChatTS-7B on TSR at layers 0, 21, and 35 for **(a)** No-CoT, **(b)** explicit chain-of-thought, and **(c)** latent reasoning. Color gives attention relative to the uniform level, red above and blue below, and the bars on the two axes mark the **time series**, **question**, **reasoning**, and **answer** spans.

A.6 Attention on the Time Series

A model can score well on multiple-choice temporal QA by exploiting linguistic priors over the answer choices instead of reading the signal, and accuracy alone says little about which part of the prompt each format actually works over. To shed light on this, we investigate the attention weight distributions during generation.

Setting. We run the attention analysis in this section on ChatTS-7B, a concatenation backbone, since SLIP and OpenTSLM inject the series through cross-attention, where the allocation is not measurable. We take a representative TSR example and average λ_{ts} over the reasoning-token queries of each format, namely the answer token for No-CoT, the chain-of-thought span for CoT, and the latents for latent reasoning.

Attention Maps. We visualize the model's layer-wise log-scaled attention vectors following the protocol of Xiao et al. (2024). A row is a generated token and a column is a token it can look back at, so the maps are lower triangular. The time series column is brightest under latent reasoning at the input and fades in the later layers, so the signal is absorbed early and the deepest layers work over text. The reasoning column is wide in panel (b) and almost absent in panel (c), so the mass

Table 8: **Attention on the time series** with ChatTS-7B. Allocation λ_{ts} (%) is the share of a reasoning token's attention that lands on the time series tokens, at the first, middle, and last layers.

Layer	No-CoT	CoT	Lapras
0 (first)	2.95	3.39	17.35
21 (middle)	11.18	4.65	6.54
35 (last)	0.72	0.27	1.21

that leaves the time series column under CoT is the mass its own trace takes. Latent reasoning has no such trace to fall back on and therefore keeps the series in view through the depths where the computation happens.

Attention Allocation. To quantify the allocation, we follow Chen et al. (2024) and measure *the total attention score one type of token receives in one layer*. Let $\alpha_{qj}^{(\ell)}$ be the post-softmax attention that a generated token q pays to a key j at layer ℓ, averaged over heads, and let S be the set of time series tokens. The allocation of the series is

$$\lambda_{\mathrm{ts}}^{(\ell)} = \sum_{j \in S,\, j<q} \alpha_{qj}^{(\ell)}, \qquad \text{where} \quad \sum_{j \le q} \alpha_{qj}^{(\ell)} = 1.$$

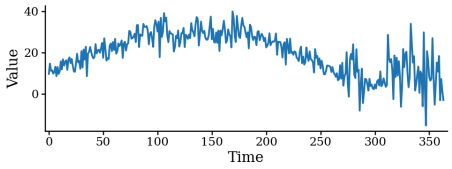
Since each row sums to one, λ_{ts} is the fraction of a token's attention budget that goes to the signal, and the rest goes to the other spans it can look back at.

As Table 8 shows, latent reasoning allocates the most attention to the time series in the early layers, where it has direct access to the series, and we read this as the model working out which part of the series matters given the context and the question. Once the series has been read, the later layers can move on to the high-level reasoning step, so the allocation drops. No-CoT behaves differently and peaks in the middle layers. We hypothesize that the model leans on the text prior in the early layers and only looks back at the series once it has to commit to an answer. CoT is flat across depth and stays below the other two formats almost everywhere. We attribute this to the verbalization bottleneck (Figure 9), which Lapras is designed to avoid.

A.7 ADDITIONAL VISUALIZATION

Question

A city's energy grid operator must decide on the allocation of emergency backup generators based on whether the temperature data indicates a prolonged period of extreme cold stress requiring maximum heating output, or if the data reflects a return to normal seasonal baselines. Based on the temperature trend shown in the dataset, which operational decision regarding the city's heating infrastructure is most justified.



A, Deactivate heating systems entirely as the data indicates a dangerous heatwave has occurred.

B, Postpone all maintenance schedules indefinitely due to unpredictable temperature fluctuations.

C, Maintain standard heating capacity as the temperatures have remained within the expected seasonal range.

D, Activate emergency backup generators immediately to prevent grid failure due to sustained extreme cold demand.

Reference CoT

The time series fluctuates between approximately **-10 and 40** throughout the entire duration, showing **no sustained spike above the baseline**. The signal exhibits **high-frequency noise but lacks a prolonged period of extreme elevation** that would indicate a persistent anomaly.
The context requires deciding between activating emergency generators for "sustained extreme cold stress" or maintaining standard operations for "normal seasonal baselines."
Since the observed behavior explicitly states there is "no sustained spike above baseline," the data **does not support the existence of a prolonged extreme event** requiring emergency intervention.
Therefore, the grid operator should **treat the conditions as within the expected range and maintain standard capacity** rather than activating expensive emergency backups.

Answer: C

Explicit Reasoning

The time series exhibits a **distinct and sustained upward trend starting around index 50, reaching a peak near index 120, and maintaining elevated levels significantly above the initial baseline until approximately index 220.**
The context specifies that this dataset covers the period of extreme cold stress. **The observed behavior—a prolonged elevation of the signal values—indicates a period of high energy consumption (thermal load) well above the normal baseline, which corresponds to the "extreme cold stress" scenario.**
Since the values are high and sustained, the grid is under continuous, severe stress.
Therefore, to prevent a total system failure due to this specific prolonged demand, **the most justified operational decision is to activate emergency backup generators to meet the excess thermal load.**

Answer: D

Figure 9: **Verbalization bottleneck in CoT reasoning.** An operational decision over a temperature series with four choices. *CoT* (bottom right) runs a CoT trace that first verbalizes the signal as text, then reasons from that textual summary rather than the original representation, and reaches an incorrect final answer. Spans are colored grounded, correct but insufficient, and incorrect.

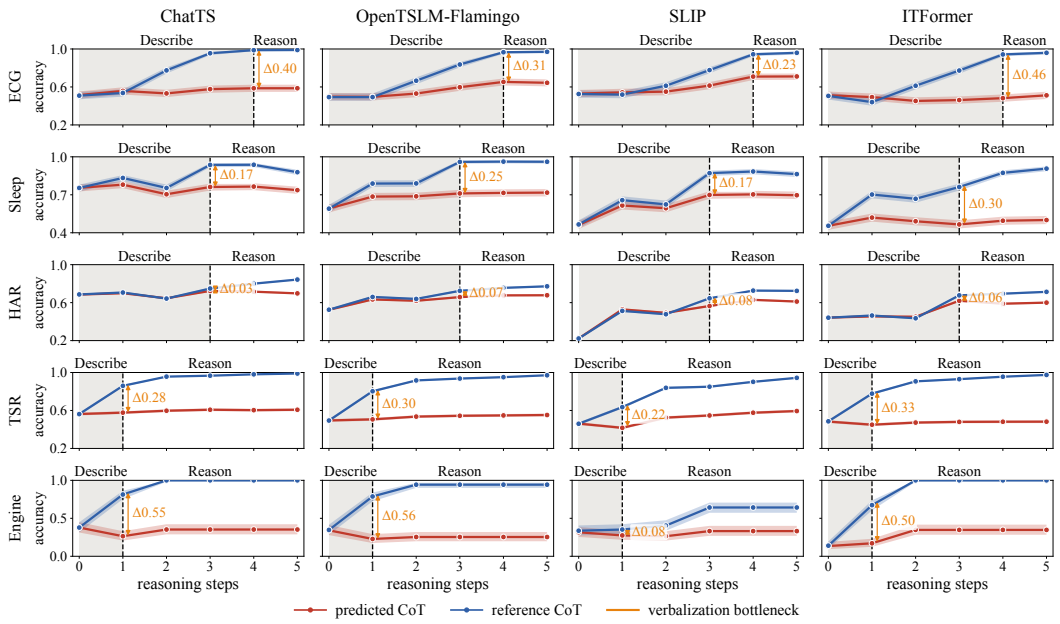


Figure 10: **Early-answering accuracy under predicted and reference CoT.** Accuracy of a frozen CoT-SFT model answering from only the first k CoT sentences, using either its own predicted CoT or the reference annotation. The verbalization bottleneck Δ is the accuracy difference between the two at the end of the *describe* stage. Shaded regions mark the *describe* and *reason* stages.

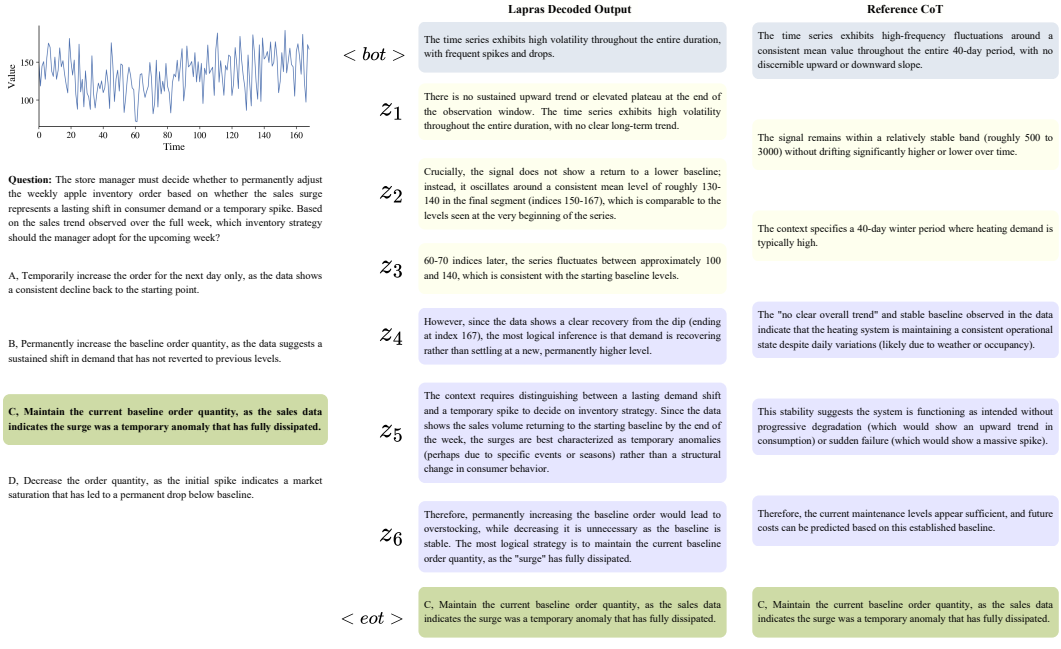


Figure 11: **Case study of Lapras's interpretability**, obtained by decoding each of the six continuous thoughts $z_1, \ldots, z_6$ into text and placing it beside the explicit chain. The decoded steps progress from a global description of the series, through local descriptions of individual segments, to inferences that combine them, and finally to the answer. Highlight colors match the box fills in the figure.