

When Reasoning Meets Structure: Understanding Reasoning Strategies for Structured Analysis

Xin Qiu¹ Jialin Gan^{1,2*}

¹Zhejiang University

²Zhejiang University-University of Illinois Urbana-Champaign Institute

qiuxin688@gmail.com

Abstract

Recent studies have explored adapting LLMs to time series tasks by leveraging language reasoning paradigms. However, it remains unclear whether reasoning strategies are effective across time series tasks with diverse temporal structures. In this work, we systematically investigate this practice using five general reasoning strategies across 12 LLMs and 20 public datasets, together with a self-constructed benchmark of approximately 18k QA pairs. Our results show that general reasoning strategies are not universally beneficial: their effectiveness depends on alignment with task-specific temporal decision structures and output spaces, rather than on model scale or complexity. We further observe that prior information and common time series pre-processing techniques exhibit strong task-dependent effects. Based on these findings, we propose three time-aware reasoning strategies that explicitly model temporal regimes, latent dynamics, and output uncertainty. Across four tasks, they consistently outperform generic reasoning strategies, and enable reasoning to better benefit from larger model scales and multi-sampling. Our study highlights the importance of temporal structure awareness in designing reliable reasoning strategies. **Our code and datasets.**

1 Introduction

Time series data are pervasive across a wide range of real-world applications (Dau et al., 2019; Ismail Fawaz et al., 2019; Dalla et al., 2026; Liu et al., 2026a). Traditional research on time series analysis has relied on statistical methods or carefully designed models, with a strong emphasis on modeling numerical patterns (Torres et al., 2021; Zhou et al., 2023b). In recent years, large language models (LLMs) have demonstrated remarkable capabilities in language reasoning, which has motivated growing interest in a fundamental

question: can LLMs be effectively applied to time series reasoning tasks, and if so, how should they be utilized?

Recent studies have explored transforming time series into natural language descriptions, enabling LLMs to perform downstream tasks (Xue and Salim, 2023; Kong et al., 2025; Qiu and Liu, 2026). They often draw inspiration from successful practices in language reasoning, such as direct prompting and multi-step reasoning strategies (Zhou and Yu, 2025; Liu et al., 2024a, 2025b). However, most existing methods implicitly rely on an assumption that has yet to be rigorously validated in the context of time series analysis: does introducing more elaborate reasoning processes improve performance? Moreover, as model scale increases, do reasoning chains continue to benefit time series tasks?

In contrast to language, time series data are governed by temporal structures and dynamic constraints (Herzen et al., 2022; Zeng et al., 2023). It raises a fundamental concern as to whether language-based reasoning paradigms, can faithfully respect such temporal constraints. In particular, for interpretability-oriented or open-ended settings, fluent reasoning narratives may obscure violations of underlying temporal consistency. Another unresolved issue concerns the task-specific sensitivity of reasoning strategies. Although forecasting, classification, anomaly detection, and open-ended question answering all operate on time series, they rely on temporal information in markedly different ways. This raises a key question: can a unified reasoning strategy remain effective across diverse time series tasks? In addition, the role of prior information, such as timestamps and background description, and time series pre-processing techniques (e.g., decomposition and normalization) remains poorly understood across different reasoning paradigms (Jin et al., 2024; Pan et al., 2024; Liu et al., 2025a).

In this work, we conduct a systematic study of

*Corresponding author.

five general reasoning strategies across six model families and twelve LLMs, using 20 publicly available real-world datasets comprising $\sim 18k$ QA pairs. The evaluation covers three types of numerical tasks and three types of QA tasks. We find that reasoning strategies are not universally beneficial: their effectiveness primarily depends on whether the reasoning form aligns with the task’s decision structure and output space, rather than on model scale or the complexity of reasoning steps. Moreover, time series prior information and pre-processing exhibit strong task dependence. We therefore propose three time-aware reasoning strategies that explicitly model temporal regime segmentation, latent dynamics, and distributional outputs, respectively, addressing the lack of temporal structure awareness in generic reasoning strategies. Our designs improve the stability and reliability of LLMs in time series analysis. Across four task categories, they consistently outperform generic reasoning strategies and mitigate the mismatch between reasoning strategies and time series tasks, enabling reasoning to better benefit from larger model scales and techniques.

Our contributions can be summarized as follows:

1. We empirically show that generic reasoning is fragile for time series. Across models, tasks, and scales, reasoning helps only when its structure matches the temporal decision space.
2. We identify time series-specific sources of reasoning failure. Prior information and common pre-processing techniques introduce strong task-dependent effects, explaining why reasoning strategies succeed in some settings but fail in others.
3. We provide a principled fix via time-aware reasoning. Our three time-aware designs restore temporal alignment, yielding consistent gains, improved stability.

2 Related Work

Reasoning Strategy for LLMs. Reasoning strategies for LLMs has rapidly advanced and demonstrated strong effectiveness across multiple modalities and challenging tasks (Zhang et al., 2024; Wang et al., 2024a; Yang et al., 2025). Starting from Direct Prompting, more sophisticated reasoning strategies such as Chain-of-Thought (Wei et al., 2022), Self-Consistency (Wang et al., 2023),

Least-to-Most (Zhou et al., 2023a), and Tree-of-Thoughts (Yao et al., 2023) have been proposed to enhance complex problem-solving capabilities. They have been shown to improve performance in medical applications (e.g., clinical QA (Wu et al., 2025)), vision tasks (e.g., visual reasoning (Dong et al., 2025; Li et al., 2025)), and mathematical problem solving (e.g., step-by-step proofs (Yu et al., 2025; Mukherjee et al., 2025)). In the time series domain, LSTPrompt (Liu et al., 2024a) and TimeReasoner (Cheng et al., 2026) have also adopted reasoning-enhanced approaches to strengthen performance. However, existing reasoning strategy for time series are often evaluated in isolated settings, providing limited insights into their effectiveness across different task formulations (Ye et al., 2025).

LLM-Based Time Series Analysis. A growing body of work has investigated the use of LLMs for time series analysis tasks such as forecasting (Gruver et al., 2023; Liu et al., 2024b), classification (Tao et al., 2024; Jing et al., 2025), anomaly detection (Liu et al., 2025c) and QA (Kong et al., 2025; Gan et al., 2026). For example, CrisisTS combines textual crisis reports with meteorological time series information to enhance multilingual urgency assessment, outperforming text-only approaches (Meunier et al., 2025). Time-MQA (Kong et al., 2025) supports multi-task time series QA by combining language reasoning with temporal data, and ITFormer (Wang et al., 2025b) presents an end-to-end framework that integrates time series modeling with language generation to enable complex multimodal QA in engine maintenance scenarios. In heterogeneous and long-horizon time series scenarios, the reliance of these methods on language-driven reasoning also indicates that effectively leveraging the potential of LLMs for time series analysis is important.

3 Preliminary

3.1 Task Definition

Given a time series of length L , $\mathbf{X} = \{\mathbf{x}_t\}_{t=1}^L$, we consider the following four time series tasks.

Classification (Cla) Task. The goal of time-series classification is to assign the input sequence $\mathbf{X}$ to a predefined label space $\mathcal{Y}$ by learning a mapping function $f(\cdot)$: $\hat{y} = f(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_L)$, $\hat{y} \in \mathcal{Y}$.

Anomaly Detection (Ano) Task. Given the input sequence $\mathbf{X}$, anomaly detection aims to identify whether each time step corresponds to a normal

or anomalous state: $\hat{a}_t = f(\mathbf{x}_t \mid \mathbf{x}_{1:t-1}), \hat{a}_t \in \{0, 1\}, t = 1, \dots, L$.

Forecasting (Fore) Task. Time-series forecasting aims to predict future values over a horizon of length H based on historical observations: $\hat{\mathbf{x}}_{L+1:L+H} = f(\mathbf{x}_{1:L})$.

Question Answering (QA) Task. Given the time series $\mathbf{X}$ and a question q , the goal is to generate a textual answer a that is consistent with the underlying temporal dynamics: $\hat{a} = f(q \mid \mathbf{x}_{1:L})$.

3.2 General Reasoning Strategies

We consider five general strategies. *Direct Prompting (DiP)* directly queries the model with the task instruction. *Chain-of-Thought (CoT)* (Wei et al., 2022) encourages step-by-step reasoning by prompting the model to generate intermediate reasoning steps. *Least-to-Most (L2M)* (Zhou et al., 2023a) decomposes a complex problem into a sequence of simpler sub-problems and solves them progressively. *Tree-of-Thoughts (ToT)* (Yao et al., 2023) explores multiple reasoning paths in parallel and selects among them based on intermediate evaluations. *Self-Refine (SRF)* (Madaan et al., 2023) iteratively improves an initial answer by generating self-feedback and refining the outputs.

3.3 System Configuration

Models. We select Qwen-2.5-(7B, 14B, 32B, and 72B) Instruct  (Yang et al., 2024), LLaMa-3-(8B and 70B) Instruct  (Grattafiori et al., 2024), Ministral-3-(3B, 8B and 14B) Instruct  (Mistral AI, 2024) and Phi-4-mini Instruct  (Abouelenin et al., 2025) to examine scaling effects within a single model family as well as cross-architecture comparisons. We also include Gemini-2.0-Flash  (Google DeepMind, 2024) and GPT-4o  (OpenAI, 2024) as closed-source models to provide a more comprehensive view.

Datasets. For the three types of numerical tasks, we evaluate our select a total of 20 publicly available real datasets. Specifically, for **Cla** task, we use the RWC, TEE, ECG, EMG, CTU, and HAR datasets, covering diverse time series types. For **Ano** task, we adopt the KPI, WSD, Yahoo, NAB, and NASA (M, S) datasets. For **Fore** task, we consider the ETTh (1,2), ETTm (1,2), AQShunyi, NASDAQ, NN5, and Wind datasets. For the **QA** task, we design four types of questions: *descriptive questions*, *relational questions*, *explanatory questions*, and *inferential questions*. The resulting

QA dataset comprises 4,965 true-false questions (TFQs), 5,877 multiple choice questions (MCQs), and 6,844 open-ended questions (OPQs).

Evaluation Protocol. For the three numerical tasks, the **Cla** task is evaluated using accuracy. The **Ano** task covers both point anomalies and interval anomalies, and is evaluated with the F1-score. The **Fore** task is evaluated using mean absolute error (MAE), which reflects the deviation between predicted sequences and the ground truth. In **Fore** task, the input length and horizons are set to $\{L: 192, H: 192\}$, except for NASDAQ, where $\{L: 48, H: 48\}$ is used. For the **QA** tasks, we evaluate TFQs and MCQs using accuracy, as each question has a single correct answer, and OPQs are assessed with ROUGE-L (Lin, 2004).

Implement Details. We deploy open-source LLMs using vLLM (Kwon et al., 2023), with top- p sampling set to 0.9 and temperature set to 0.7. Closed-source models are accessed via their official APIs with default configurations. For iterative reasoning strategies, including *ToT* and *SRF*, the reasoning process is performed over N rounds. Specifically, *ToT* evaluates N candidate paths and selects the best one, while *SRF* iteratively refines the prediction from the final round. For non-iterative reasoning strategies, including *DiP*, *CoT*, and *L2M*, we generate N independent responses in parallel and determine the final answer via majority voting. Throughout all experiments, unless otherwise specified, N is set to 5.

4 Exploring General Reasoning Strategies for Time Series Analysis

This section takes a holistic view of how general reasoning strategies operate in time series analysis.

4.1 Research Questions (RQs)

RQ1: Are general reasoning strategies universally beneficial for time series analysis?

General reasoning strategies have proven effective in language and symbolic tasks (Xu et al., 2024; Ranaldi et al., 2025). However, time series analysis demands sensitivity to temporal structure and numerical precision. Without explicitly encoding such structures, these reasoning strategies may not fully realize their advantages. Motivated by this gap, we evaluate whether general reasoning strategies constitute a universally effective inductive bias for time series analysis.

RQ2: Do more samples better leverage reasoning strategies?

Many reasoning strategies can be combined with multiple sampling to improve answer quality (Wan et al., 2025). Iterative methods perform multi-round refinement or explore multiple paths, and non-iterative strategies may benefit from parallel sampling. However, it remains unclear whether increasing sampling times improves the effectiveness of reasoning strategies in time series analysis.

RQ3: Does incorporating prior information benefit reasoning strategies?

Some studies suggest that prior information provides models with additional contextual cues (Yang, 2025; Kong et al., 2025). Timestamp encodes temporal positions, facilitating reasoning aligned with temporal structure, and background descriptions offer high-level semantic context, such as domain, scenario, or system state, reducing reasoning ambiguity. However, whether such priors can be translated into improved performance remains unclear.

RQ4: Does time series pre-processing help reasoning strategies perform better?

Time series pre-processing techniques are commonly regarded as helpful for highlighting salient information (Pan et al., 2024; Kreuzer et al., 2025). In particular, decomposition separates the original sequence into trend, seasonal, and residual components, revealing structural patterns, and normalization removes scale variations and unit discrepancies. Although they simplify the input space, whether they can reliably enhance reasoning strategies remains an open question.

RQ5: Do larger models benefit more from reasoning strategies?

Previous studies have shown that scaling model parameters yields greater performance gains under more complex reasoning strategies (Wang et al., 2025a; Balachandran et al., 2025). However, in time series analysis, reasoning is constrained by numerical modeling and temporal dependencies, and whether larger models can more effectively leverage reasoning strategies remains under-explored.

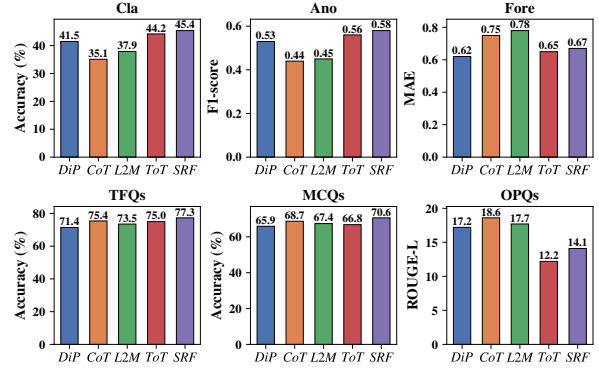


Figure 1: Performance comparison of five reasoning strategies across six tasks. Accuracy ($\uparrow$, %), F1-score ($\uparrow$), ROUGE-L ($\uparrow$), and MAE ($\downarrow$) are reported as averages over six models.

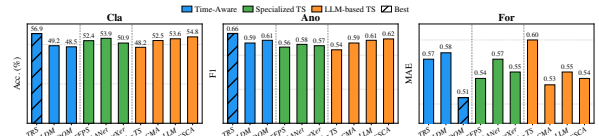


Figure 2: Comparison with baselines across three tasks.

This relationship determines when scaling truly improves analytical capability.

4.2 Empirical Results

RQ1. We select four open-source models, including Qwen-2.5-7B Instruct, LLaMA-3-8B Instruct, Ministral-3-3B Instruct, and Phi-4-mini Instruct, together with two closed-source models, Gemini-2.0-Flash and GPT-4o, resulting in a total of six models. They are evaluated on four time series analysis tasks under five general reasoning strategies, which include *DiP*, *CoT*, *L2M*, *ToT*, and *SRF*.

As shown in Fig. 1, on the Cla and Ano tasks, *ToT* and *SRF* generally improve performance, whereas *CoT* and *L2M* tend to degrade it. In contrast, for OPQs task, the trend is reversed. For TFQs and MCQs tasks, all four reasoning strategies lead to consistent performance gains. However, none of the strategies improves performance on the Fore task; instead, they result in higher MAE.

As shown in Fig. 2, we compare our time-aware strategies with specialized time series models (TFPS (Sun et al., 2026), TSLANet (Eldele et al., 2024) and TimeXer (Wang et al., 2024b)) and LLM-based time series baselines (LLM-TS (Chen et al., 2025), TimeCMA (Liu et al., 2025a), SE-LLM (Liu et al., 2026b) and FSCA (Hu et al., 2025)). The results show that an appropriate reasoning strategy can outperform dedicated time series architectures and existing time series LLMs.

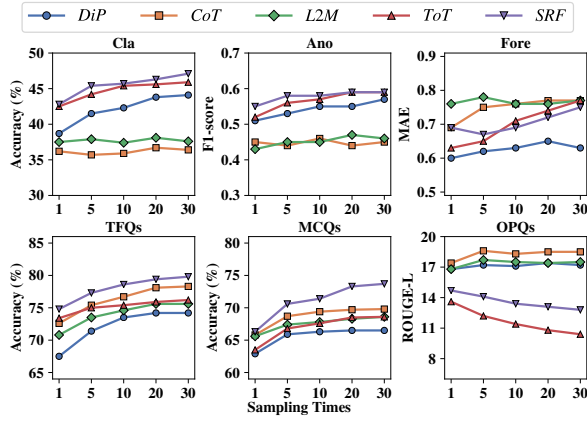


Figure 3: Average performance of general reasoning strategies across six models with different sampling times.

Takeaway of RQ1: Non-iterative strategies (*CoT*, *L2M*) often underperform *DiP* on Cla, Ano, and Fore tasks, but benefit QA tasks, but iterative strategies (*ToT*, *SRF*) improve Cla, Ano, TFQs, and MCQs tasks, yet perform poorly on Fore and OPQs tasks.

RQ2. We gradually increase the sampling times from 1 to 5, 10, 20, and 30 to examine how the performance of reasoning strategies varies across tasks, while keeping all other settings fixed.

As shown in Fig. 3, for the Cla, Ano, and OPQs tasks, increasing sampling times leads to performance improvements for *DiP*, *CoT*, and *L2M*. For the TFQs and MCQs tasks, all reasoning strategies benefit from increased sampling. In the remaining cases, increasing sampling times yields no clear gains or even degrades performance.

Takeaway of RQ2: Increasing sampling times tends to benefit *DiP*, *CoT*, and *L2M* on Cla, Ano, and OPQs tasks, and all strategies on TFQs and MCQs tasks, with limited or negative effects elsewhere.

RQ3. For the Cla, Ano, and Fore tasks, inputs include the time series together with prior information (timestamps and background descriptions, BDs). We evaluate reasoning strategies on the six models by ablating these components individually.

As shown in Fig. 4, the Cla and Ano tasks are evaluated using Accuracy and F1-score, where higher values indicate better performance. Removing BDs leads to performance degradation, but timestamps have a relatively smaller impact. In contrast, the Fore task is evaluated using MAE, where

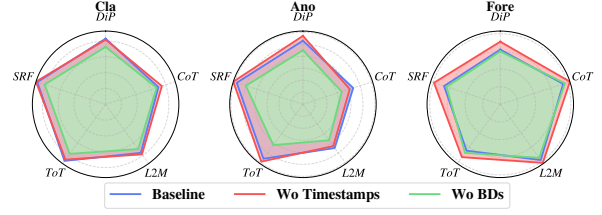


Figure 4: Impact of prior information ablation on reasoning strategies, averaged over six models.

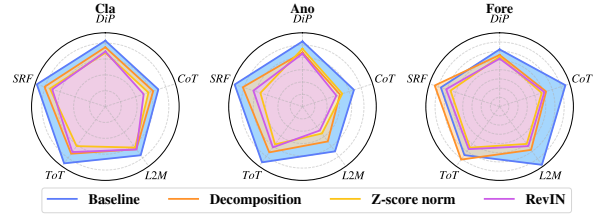


Figure 5: Impact of time series pre-processing on reasoning strategies, averaged over six models.

larger values indicate higher errors, and timestamps have a much greater impact than BDs.

Takeaway of RQ3: Cla and Ano rely more on background descriptions, but Fore is more sensitive to timestamps.

RQ4. In addition to prior information, we examine the impact of time series pre-processing on reasoning strategies using three approaches: time series decomposition, Z-score normalization (Hastie et al., 2009), and RevIN (Kim et al., 2021).

Results show that pre-processing has little benefit for the Cla and Ano tasks; in particular, z-score normalization and RevIN often lead to severe performance degradation. However, all methods improve performance on the Fore task, with the gains being most pronounced for *CoT* and *L2M*, as shown in Fig. 5.

Takeaway of RQ4: Pre-processing is largely ineffective for Cla and Ano tasks, often harmful with z-score normalization and RevIN, but consistently benefits Fore, especially under *CoT* and *L2M*.

RQ5. Using nine models from three families, Qwen-2.5 Instruct (7B, 14B, 32B, 72B) 🦙, LLaMA-3 Instruct (8B, 70B) 🦙, and Ministral-3 Instruct (3B, 8B, 14B) 🦙, we investigate whether larger models benefit more from reasoning strategies.

As shown in Fig. 6, performance on TFQs and

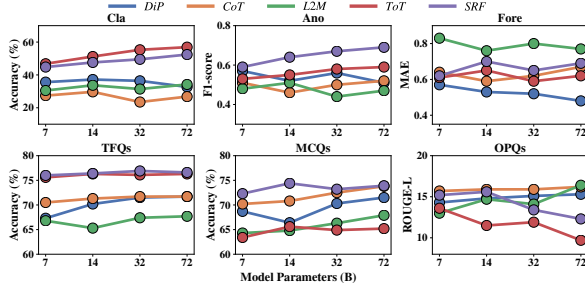


Figure 6: Performance trends of different reasoning strategies as model parameters increases within the Qwen-2.5 Instruct.

MCQs tasks generally improves with increasing model size across all reasoning strategies. For Cla and Ano tasks, *ToT* and *SRF* exhibit stable gains, while other strategies show marginal improvements or even performance degradation in some cases.

Takeaway of RQ5: Across all cases, model scaling tends to benefit TFQs and MCQs tasks across strategies, while only *ToT* and *SRF* show gains on Cla and Ano tasks.

4.3 Understanding When and Why Reasoning Strategies Help

Reasoning strategies act as task-specific inductive biases rather than universal performance boosters. A strategy improves performance only when its induced reasoning structure aligns with task requirements; otherwise, additional reasoning may introduce noise and degrade decision-making.

The effectiveness of reasoning is governed by the task’s output space and decision structure. Reasoning benefits tasks with discrete, finite answer spaces (e.g., Cla and QA tasks) by enabling comparison and calibration of candidate decisions, whereas for continuous regression tasks, the lack of clear decision boundaries makes reasoning harder to constrain and prone to error amplification.

Iterative reasoning is more robust under decision uncertainty, while non-iterative reasoning is more susceptible to noise amplification. Iterative strategies such as *ToT* and *SRF* improve stability by exploring multiple reasoning paths or applying self-refinement, whereas non-iterative strategies like *CoT* and *L2M* rely on one-shot reasoning and are more vulnerable.

Reasoning strategies are highly sensitive to input representations, with semantic and temporal priors playing task-dependent roles. Semantic priors benefit Cla and Ano tasks by providing high-

level context, whereas Fore tasks rely more on temporal alignment. Pre-processing techniques that favor Fore task may distort relative structures in Cla and Ano tasks, potentially weakening reasoning effectiveness.

Model scaling amplifies reasoning patterns that align with task structure, but cannot compensate for mismatched reasoning strategies. Empirical results show that larger models yield consistent gains for discrete decision tasks, while in tasks such as Cla and Ano tasks, only iterative strategies (e.g., *ToT* and *SRF*) scale reliably. It indicates that scaling primarily strengthens task-aligned reasoning modes, rather than rescuing ineffective ones, making alignment between reasoning form and task demands the key factor governing performance.

5 When Reasoning Meets Temporal Structure

Sec. 4 reveals substantial instability in the effectiveness of general reasoning strategies for time series analysis tasks. It may partly stem from the fact that most reasoning strategies are primarily designed for general language tasks and lack explicit structural awareness of key time series properties (Leppich et al., 2025; Zhang et al., 2025).

5.1 Time-Aware Reasoning Strategies

Temporal Regime Segmentation (TRS). Many time series tasks are governed by stage-wise dynamics, where mixing heterogeneous regimes. A time series can be decomposed into multiple segments with distinct temporal characteristics:

$$\mathbf{x}_{1:T} = \bigcup_{k=1}^K \mathbf{x}_{\tau_{k-1}+1:\tau_k}, 0 = \tau_0 < \dots < \tau_K = T,$$

where τ_k denotes latent change points that separate different dynamic stages. Segment-wise reasoning results are then aggregated:

$$y = g(f(\mathbf{x}_{1:\tau_1}), \dots, f(\mathbf{x}_{\tau_{K-1}+1:T})), \quad (1)$$

where $f(\cdot)$ denotes the segment-level reasoning function and $g(\cdot)$ aggregates reasoning outcomes.

Latent Dynamics Modeling (LDM). Reasoning over individual observations may obscure long-range temporal dependencies, whereas latent dynamics provide a compact abstraction of system evolution. A time series is modeled as a latent dynamical process, where observations are generated from evolving hidden states:

$$z_t = h(z_{t-1}, u_t), x_t = g(z_t) + \epsilon_t,$$

Table 1: Comparison of reasoning strategies on time-series analysis tasks, reported as the average performance across six models. **Bold** and underlined values indicate the best and second-best results for each task, respectively.

Task	Cla	Ano	Fore	TFQs	MCQs	OPQs
Metric	Acc.↑(%)	F1↑	MAE↓	Acc.↑(%)	Acc.↑(%)	ROUGE-L↑
General Reasoning Strategies						
<i>DiP</i>	41.5	0.53	0.62	71.4	65.9	17.2
<i>CoT</i>	35.1	0.44	0.75	75.4	68.7	18.6
<i>L2M</i>	37.9	0.45	0.78	73.5	67.4	17.7
<i>ToT</i>	44.2	0.56	0.65	75.0	66.8	12.2
<i>SRF</i>	45.4	0.58	0.67	<u>77.3</u>	70.6	14.1
Time-Aware Reasoning Strategies						
<i>TRS</i>	56.9	0.66	<u>0.57</u>	75.8	72.6	19.8
<i>LDM</i>	<u>49.2</u>	0.59	0.58	85.5	77.3	23.4
<i>DOM</i>	48.5	<u>0.61</u>	0.51	81.4	<u>74.0</u>	<u>21.5</u>

where z_t denotes the latent system state at time t , u_t represents contextual influences, and ϵ_t captures observation noise. The final output is derived from the entire latent trajectory: $y = \psi(\{z_t\}_{t=1}^T)$.

Distributional Output Modeling (DOM). Time series analysis task often involves noise, non-stationarity, and limited observations, making a single deterministic output insufficient to reflect the model’s uncertainty. *DOM* represents the model output in a distributional form:

$$y \sim p(y \mid \mathbf{x}_{1:T}), \text{Unc}(y) = \mathcal{U}(p(y \mid \mathbf{x}_{1:T})),$$

where $\text{Unc}(\cdot)$ denotes an uncertainty measure. During inference, we rank candidate outputs by jointly considering expected utility and uncertainty:

$$y^* = \arg \max_y \mathbb{E}[U(y)] - \text{Unc}(y), \quad (2)$$

where $U(\cdot)$ instantiates the task-specific utility (e.g., likelihood of the correct option for QA tasks).

5.2 Empirical Evaluation

Based on the three time-aware reasoning perspectives introduced above, we conduct an empirical evaluation to assess their effectiveness. We evaluate them across six models (Qwen-2.5-7B Instruct, LLaMA-3-8B Instruct, Ministral-3-3B Instruct, Phi-4-mini Instruct, Gemini-2.0-Flash and GPT-4o) and six categories of tasks.

Tab. 1 reports the average results across six models. Compared with the best-performing general reasoning strategy, the best time-aware reasoning strategy achieves relative performance improvements of 25.3%, 13.8%, 17.7%, 10.6%, 9.5%, and 25.8% on the six tasks, respectively.

Further analysis reveals that the advantages of different time-aware reasoning strategies exhibit

strong task dependency. *TRS* achieves the most pronounced performance gains on Cla and Ano tasks. *TRS* effectively mitigates decision interference caused by mixing heterogeneous temporal regimes. In contrast, *LDM* demonstrates strong performance on Fore task and QA tasks (TFQs, MCQs, and OPQs). *LDM* better aligns with the decision structure of Fore tasks, leading to a substantial reduction in MAE. Moreover, in QA tasks that require integrating multiple temporal cues, latent dynamic modeling helps suppress the influence of local noise. *DOM* exhibits more uniform advantages across QA tasks and achieves the best performance on OPQs. By modeling a distribution over multiple candidate reasoning outcomes and accounting for uncertainty, *DOM* improves the reliability of answer selection.

5.3 Scalability of Time-Aware Strategies

Multi-Sampling Scalability. Although increasing the sampling times N can benefit certain tasks, RQ2 shows that such gains are not universal. This suggests that the effectiveness of multi-sampling depends on whether the underlying reasoning strategy captures task-relevant time series characteristics, motivating the question of whether multi-sampling is more effective when applied to time-aware reasoning strategies.

To answer this question, we examine the effect of multi-sampling by varying the sampling budget $N \in \{1, 5, 10, 20, 30, 50\}$. For non-iterative strategies such as *TRS* and *LDM*, we generate N independent reasoning outputs and aggregate them using majority voting. For the strategy *DOM*, multi-sampling naturally induces a distribution over candidate outputs, where increasing N yields a more accurate estimate of output uncertainty and utility. Due to space limitations, we report the mean performance averaged over the six models.

Results of Fig. 7 show that, unlike general reasoning strategies, time-aware reasoning strategies can reliably translate larger sampling budgets into stable performance improvements, exhibiting convergent behavior. This confirms their superior sampling scalability for time series analysis tasks.

Model Scaling Behavior. Sampling primarily characterizes the scalability of reasoning at the inference stage. Another equally important factor is model scale, which determines the upper bound of representational capacity and reasoning complexity. Therefore, *do time-aware reasoning strategies*

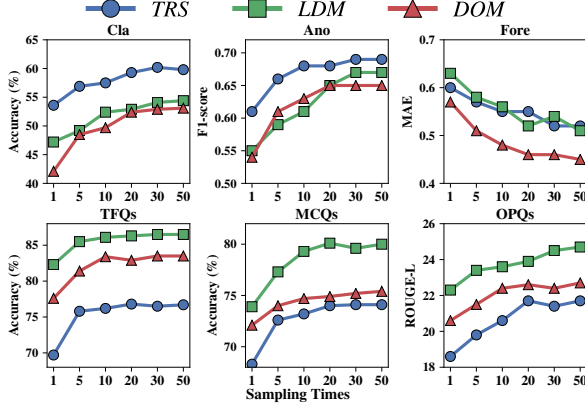


Figure 7: Average performance of time-aware reasoning strategies across six models with different sampling times.

Table 2: Performance of Qwen-Instruct family.

Task		Cla	Ano	Fore	TFQs	MCQs	OPQs
Metric		Acc.↑(%)	F1↑	MAE↓	Acc.↑(%)	Acc.↑(%)	ROUGE-L↑
TRS	7B	52.3	0.69	0.55	77.1	74.5	21.3
	14B	53.7	0.69	0.51	78.4	72.6	21.7
	32B	54.2	0.71	0.46	80.6	76.4	23.5
	72B	55.1	0.74	0.48	82.1	77.9	23.2
LDM	7B	47.4	0.61	0.53	80.6	75.8	20.9
	14B	48.6	0.63	0.51	82.6	76.5	20.7
	32B	50.3	0.65	0.48	83.1	77.1	21.5
	72B	50.9	0.66	0.46	83.5	77.6	22.4
DOM	7B	46.1	0.64	0.46	76.3	83.2	20.0
	14B	46.9	0.65	0.44	77.5	83.5	20.6
	32B	48.3	0.68	0.41	79.1	84.6	21.4
	72B	49.8	0.68	0.42	78.4	84.9	21.9

amplify their advantages as model size increases?

We analyze how the effectiveness of three time-aware reasoning strategies varies with model scale by evaluating nine models from three families: Qwen-2.5 Instruct (7B, 14B, 32B, 72B) 🦑, LLaMA-3 Instruct (8B, 70B) 🦙, and Ministral-3 Instruct (3B, 8B, 14B) 🏠. As shown in Tab. 2, 3, 4, *TRS*, *LDM*, and *DOM* demonstrate clear performance improvements across different tasks and model families as the model size increases. These results indicate that time-aware reasoning strategies can reliably translate increased model scale into effective reasoning on time series tasks.

6 Discussion

Limits of General Reasoning in Time Series Analysis. Their effectiveness varies across tasks, sampling budgets, input priors, and model scales, and may even degrade performance in certain settings due to a mismatch between language-oriented reasoning mechanisms and the temporal structure of time series data. In contrast, time-aware reasoning strategies are designed to account for temporal characteristics, resulting in more stable and reliable

Table 3: Performance of LLaMA-Instruct family.

Task		Cla	Ano	Fore	TFQs	MCQs	OPQs
Metric		Acc.↑(%)	F1↑	MAE↓	Acc.↑(%)	Acc.↑(%)	ROUGE-L↑
TRS	8B	50.7	0.64	0.62	71.5	65.3	17.6
	70B	53.3	0.68	0.57	76.3	70.3	20.6
LDM	8B	48.6	0.54	0.63	82.4	75.9	22.1
	70B	51.7	0.58	0.52	87.7	81.4	25.7
DOM	8B	44.7	0.59	0.58	77.5	80.1	18.6
	70B	48.5	0.66	0.52	81.2	84.9	22.4

Table 4: Performance of Ministral-Instruct family.

Task		Cla	Ano	Fore	TFQs	MCQs	OPQs
Metric		Acc.↑(%)	F1↑	MAE↓	Acc.↑(%)	Acc.↑(%)	ROUGE-L↑
TRS	3B	47.2	0.62	0.74	68.4	71.4	15.4
	8B	49.6	0.65	0.65	73.2	73.6	17.9
	14B	52.5	0.67	0.60	75.8	73.8	18.5
LDM	3B	44.5	0.47	0.69	72.5	73.5	18.6
	8B	46.9	0.53	0.65	76.4	74.8	18.2
	14B	48.6	0.56	0.62	77.9	74.3	19.7
DOM	3B	43.0	0.54	0.62	69.5	78.3	17.5
	8B	43.9	0.59	0.57	73.6	81.4	18.7
	14B	45.4	0.61	0.54	75.8	82.0	19.4

improvements across tasks and settings.

Time Series Reasoning Exhibits Strong Task Specificity. Different tasks vary in stage heterogeneity, long-term dependency, and uncertainty, making it unlikely for a single reasoning form to perform well across all scenarios. A more effective approach is therefore to select strategies based on temporal structure and decision requirements.

Structure Alignment Precedes Computational Scaling. Additional computational resources, such as larger sampling budgets or model scales, appear insufficient to overcome mismatches. Instead, performance gains critically depend on aligning the reasoning process with task-specific temporal characteristics and decision requirements.

7 Conclusion

Our work shows that generic reasoning strategies do not reliably transfer to TS analysis, as their effectiveness depends on alignment with task-specific temporal decision structures rather than reasoning complexity or model scale. We further reveal strong task-dependent effects of prior information and TS preprocessing. To address these limitations, we propose three time-aware strategies that incorporate temporal regimes, dynamics, and uncertainty, consistently improving performance, stability, and scalability across diverse TS tasks. Our findings highlight temporal structure awareness as a key principle for reliable LLM-based TS reasoning.

Limitations

Multi-sampling and iterative reasoning demonstrate clear advantages in improving reasoning stability, but they also incur increased computational cost during inference. This work primarily focuses on a systematic analysis of reasoning effectiveness and stability, and does not extensively investigate deployment efficiency under strict latency or resource-constrained settings. In addition, we categorize time series tasks into several representative decision forms to facilitate systematic comparison of reasoning strategies. While this categorization helps reveal overall trends, it may not fully capture hybrid or borderline task settings. Moreover, we focus on analyzing individual reasoning strategies in isolation to uncover the relationship between reasoning forms and task structures. Although combining or coordinating multiple reasoning strategies may lead to more complex reasoning behaviors, their systematic design and evaluation remain open directions for future work.

References

- Abdelrahman Abouelenin, Atabak Ashfaq, Adam Atkinson, Hany Awadalla, Nguyen Bach, Jianmin Bao, Alon Benhaim, Martin Cai, Vishrav Chaudhary, Congcong Chen, and 1 others. 2025. Phi-4-mini technical report: Compact yet powerful multimodal language models via mixture-of-loras. *arXiv preprint arXiv:2503.01743*.
- Mohammad Arashi and Mohammad Mahdi Rounaghi. 2022. Analysis of market efficiency and fractal feature of nasdaq stock exchange: Time series modeling and forecasting of stock index using arma-garch model. *Future Business Journal*, 8(1):14.
- Vidhisha Balachandran, Jingya Chen, Lingjiao Chen, Shivam Garg, Neel Joshi, Yash Lara, John Langford, Besmira Nushi, Vibhav Vineet, Yue Wu, and 1 others. 2025. Inference-time scaling for complex tasks: Where we stand and what lies ahead. *arXiv preprint arXiv:2504.00294*.
- Can Chen, Gabriel L. Oliveira, Hossein Sharifi-Noghabi, and Tristan Sylvain. 2025. LLM-TS integrator: Integrating LLM for enhanced time series modeling. *Transactions on Machine Learning Research*.
- Mingyue Cheng, Jiahao Wang, Daoyu Wang, Xiaoyu Tao, Qi Liu, and Enhong Chen. 2026. Can slow-thinking llms reason over time? empirical studies in time series forecasting. In *Proceedings of the Nineteenth ACM International Conference on Web Search and Data Mining*, pages 99–110.
- Violetta Dalla, Liudas Giraitis, and Peter CB Phillips. 2026. Testing mean stability of heteroskedastic time series. *Journal of Time Series Analysis*, 47(1):182–200.
- Hoang Anh Dau, Anthony Bagnall, Kaveh Kamgar, Chin-Chia Michael Yeh, Yan Zhu, Shaghayegh Gharghabi, Chotirat Ann Ratanamahatana, and Eamonn Keogh. 2019. The ucr time series archive. *IEEE/CAA Journal of Automatica Sinica*, 6(6):1293–1305.
- Yuhao Dong, Zuyan Liu, Hai-Long Sun, Jingkang Yang, Winston Hu, Yongming Rao, and Ziwei Liu. 2025. Insight-v: Exploring long-chain visual reasoning with multimodal large language models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 9062–9072.
- Emadeldeen Eldele, Mohamed Ragab, Zhenghua Chen, Min Wu, and Xiaoli Li. 2024. Tslanet: Rethinking transformers for time series representation learning. In *International Conference on Machine Learning*, pages 12409–12428. PMLR.
- Jialin Gan, Xin Qiu, Guangzhe Chen, and Xue Wang. 2026. Beyond uniform tokens: Adaptive compression for time series language models. *arXiv preprint arXiv:2606.13624*.
- Google DeepMind. 2024. Gemini 2.0: Flash. <https://deepmind.google/technologies/gemini/>. Model release and technical overview.
- Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.
- Nate Gruver, Marc Finzi, Shikai Qiu, and Andrew G Wilson. 2023. Large language models are zero-shot time series forecasters. *Advances in Neural Information Processing Systems*, 36:19622–19635.
- Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. 2009. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer.
- Julien Herzen, Francesco Lässig, Samuele Giuliano Piazetta, Thomas Neuer, Léo Tafti, Guillaume Raille, Tomas Van Pottelbergh, Marek Pasieka, Andrzej Skrodzki, Nicolas Huguenin, and 1 others. 2022. Darts: User-friendly modern machine learning for time series. *Journal of Machine Learning Research*, 23(124):1–6.
- Yuxiao Hu, Qian Li, Dongxiao Zhang, Jinyue Yan, and Yuntian Chen. 2025. Context-alignment: Activating and enhancing llms capabilities in time series. In *International Conference on Learning Representations*, volume 2025, pages 90696–90722.
- Kyle Hundman, Valentino Constantinou, Christopher Laporte, Ian Colwell, and Tom Soderstrom. 2018. Detecting spacecraft anomalies using lstms and non-parametric dynamic thresholding. In *Proceedings of*

- the 24th ACM SIGKDD international conference on knowledge discovery & data mining, pages 387–395.
- Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. 2019. Deep learning for time series classification: a review. *Data mining and knowledge discovery*, 33(4):917–963.
- Ming Jin, Shiyu Wang, Lintao Ma, Zhixuan Chu, James Zhang, Xiaoming Shi, Pin-Yu Chen, Yuxuan Liang, Yuan-Fang Li, Shirui Pan, and 1 others. 2024. Time-llm: Time series forecasting by reprogramming large language models. In *International conference on learning representations*, volume 2024, pages 23857–23880.
- Yuhan Jing, Bo He, Haifeng Sun, Qi Qi, Zirui Zhuang, Lei Zhang, Jianxin Liao, and JingYu Wang. 2025. Foresail: Llm sensor knowledge empowered status-guided network for multivariate time-series classification. In *Proceedings of the 33rd ACM International Conference on Multimedia*, pages 11150–11159.
- Taesung Kim, Jinhee Kim, Yunwon Tae, Cheonbok Park, Jang-Ho Choi, and Jaegul Choo. 2021. Reversible instance normalization for accurate time-series forecasting against distribution shift. In *International conference on learning representations*.
- Yaxuan Kong, Yiyuan Yang, Yoontae Hwang, Wenjie Du, Stefan Zohren, Zhangyang Wang, Ming Jin, and Qingsong Wen. 2025. 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)*, pages 29736–29753.
- Tim Kreuzer, Jelena Zdravkovic, and Panagiotis Papapetrou. 2025. Unpacking the trend: Decomposition as a catalyst to enhance time series forecasting models. *Data Mining and Knowledge Discovery*, 39(5):54.
- Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*, pages 611–626.
- Alexander Lavin and Subutai Ahmad. 2015. Evaluating real-time anomaly detection algorithms—the numenta anomaly benchmark. In *2015 IEEE 14th international conference on machine learning and applications (ICMLA)*, pages 38–44. IEEE.
- Robert Leppich, Michael Stenger, Daniel Grillmeyer, Vanessa Borst, and Samuel Kounev. 2025. Tsmr: A lightweight temporal feature encoding architecture for time series forecasting and imputation. *arXiv preprint arXiv:2504.18878*.
- Lingxiao Li, Yifan Wang, Xinyan Gao, Chen Tang, Xiangyu Yue, and Chenyu You. 2025. Visreason: A large-scale dataset for visual chain-of-thought reasoning. *arXiv preprint arXiv:2511.17731*.
- Zeyan Li, Nengwen Zhao, Shenglin Zhang, Yongqian Sun, Pengfei Chen, Xidao Wen, Minghua Ma, and Dan Pei. 2022. Constructing large-scale real-world benchmark datasets for aiops. *arXiv preprint arXiv:2208.03938*.
- Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.
- Changwei Liu, Hao Ren, Guoqiang Li, Haojie Ren, Xiaojun Liang, Chunhua Yang, and Weihua Gui. 2026a. Singular value decomposition-based lightweight lstm for time series forecasting. *Future Generation Computer Systems*, 174:107910.
- Chenxi Liu, Qianxiong Xu, Hao Miao, Sun Yang, Lingzheng Zhang, Cheng Long, Ziyue Li, and Rui Zhao. 2025a. Timecma: Towards llm-empowered multivariate time series forecasting via cross-modality alignment. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 18780–18788.
- Hao Liu, Zhang xiaoxing, Chun Yang, and Xiaobin Zhu. 2026b. Semantic-enhanced time-series forecasting via large language models. In *The Fourteenth International Conference on Learning Representations*.
- Haixin Liu, Chenghao Liu, and B Aditya Prakash. 2025b. A picture is worth a thousand numbers: Enabling llms reason about time series via visualization. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 7486–7518.
- Haixin Liu, Zhiyuan Zhao, Jindong Wang, Harshvardhan Kamarthi, and B Aditya Prakash. 2024a. Lst-prompt: Large language models as zero-shot time series forecasters by long-short-term prompting. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 7832–7840.
- Jun Liu, Chaoyun Zhang, Jiaxu Qian, Minghua Ma, Si Qin, Chetan Bansal, Qingwei Lin, Saravan Rajmohan, and Dongmei Zhang. 2025c. Large language models can deliver accurate and interpretable time series anomaly detection. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pages 4623–4634.
- Yong Liu, Guo Qin, Xiangdong Huang, Jianmin Wang, and Mingsheng Long. 2024b. Autotimes: Autoregressive time series forecasters via large language models. *Advances in Neural Information Processing Systems*, 37:122154–122184.
- Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegrefe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, and 1 others. 2023. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36:46534–46594.

- Romain Meunier, Farah Benamara, Véronique Moriceau, Zhongzheng Qiao, and Savitha Ramasamy. 2025. Crisists: Coupling social media textual data and meteorological time series for urgency classification. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 16082–16099.
- Mistral AI. 2024. Mistral small 3. <https://mistral.ai/news/mistral-small-3>. Model release and technical overview.
- Sagnik Mukherjee, Abhinav Chinta, Takyoun Kim, Tarun Anoop Sharma, and Dilek Hakkani Tur. 2025. Premise-augmented reasoning chains improve error identification in math reasoning with llms. In *International Conference on Machine Learning*, pages 45109–45128. PMLR.
- OpenAI. 2024. Gpt-4o. <https://openai.com/index/hello-gpt-4o/>. Model release and technical overview.
- Zijie Pan, Yushan Jiang, Sahil Garg, Anderson Schneider, Yuriy Nevmyvaka, and Dongjin Song. 2024. S2ip-llm: Semantic space informed prompt learning with llm for time series forecasting. In *Forty-first International Conference on Machine Learning*.
- Xin Qiu and Wenjie Liu. 2026. Rpgfusion: 4d radar prior-guided multi-modal fusion for 3d detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 284–294.
- Leonardo Ranaldi, Marco Valentino, and Andre Freitas. 2025. Improving chain-of-thought reasoning via quasi-symbolic abstractions. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 17222–17240.
- Hansheng Ren, Bixiong Xu, Yujing Wang, Chao Yi, Congrui Huang, Xiaoyu Kou, Tony Xing, Mao Yang, Jie Tong, and Qi Zhang. 2019. Time-series anomaly detection service at microsoft. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 3009–3017.
- Yanru Sun, Zongxia Xie, Emadeldeen Eldele, Dongyue Chen, Qinghua Hu, and Min Wu. 2026. Learning pattern-specific experts for time series forecasting under patch-level distribution shift. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.
- Xiaoyu Tao, Tingyue Pan, Mingyue Cheng, Yucong Luo, Qi Liu, and Enhong Chen. 2024. Hierarchical multimodal llms with semantic space alignment for enhanced time series classification. *ACM Transactions on Intelligent Systems and Technology*.
- SD Teddy and SK Ng. 2011. Forecasting atm cash demands using a local learning model of cerebellar associative memory network. *International Journal of Forecasting*, 27(3):760–776.
- José F Torres, Dalil Hadjout, Abderrazak Sebaa, Francisco Martínez-Álvarez, and Alicia Troncoso. 2021. Deep learning for time series forecasting: a survey. *Big data*, 9(1):3–21.
- Guangya Wan, Yuqi Wu, Jie Chen, and Sheng Li. 2025. Reasoning aware self-consistency: Leveraging reasoning paths for efficient llm sampling. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 3613–3635.
- Jian Wang, Boyan Zhu, Chak Tou Leong, Yongqi Li, and Wenjie Li. 2025a. Scaling over scaling: Exploring test-time scaling plateau in large reasoning models. *arXiv preprint arXiv:2505.20522*.
- Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*.
- Yilin Wang, Peixuan Lei, Jie Song, Yuzhe Hao, Tao Chen, Yuxuan Zhang, Lei Jia, Yuanxiang Li, and Zhongyu Wei. 2025b. Itformer: Bridging time series and natural language for multi-modal qa with large-scale multitask dataset. In *International Conference on Machine Learning*, pages 63324–63344. PMLR.
- Yu Wang, Shiwan Zhao, Zhihu Wang, Heyuan Huang, Ming Fan, Yubo Zhang, Zhixing Wang, Haijun Wang, and Ting Liu. 2024a. Strategic chain-of-thought: Guiding accurate reasoning in llms through strategy elicitation. *arXiv preprint arXiv:2409.03271*.
- Yuxuan Wang, Haixu Wu, Jiaxiang Dong, Guo Qin, Haoran Zhang, Yong Liu, Yunzhong Qiu, Jianmin Wang, and Mingsheng Long. 2024b. Timexer: Empowering transformers for time series forecasting with exogenous variables. *Advances in Neural Information Processing Systems*, 37:469–498.
- Zexin Wang, Changhua Pei, Minghua Ma, Xin Wang, Zhihan Li, Dan Pei, Saravan Rajmohan, Dongmei Zhang, Qingwei Lin, Haiming Zhang, and 1 others. 2024c. Revisiting vae for unsupervised time series anomaly detection: A frequency perspective. In *Proceedings of the ACM web conference 2024*, pages 3096–3105.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.
- Juncheng Wu, Wenlong Deng, Xingxuan Li, Sheng Liu, Taomian Mi, Yifan Peng, Ziyang Xu, Yi Liu, Hyunjin Cho, Chang-In Choi, and 1 others. 2025.

- Medreason: Eliciting factual medical reasoning steps in llms via knowledge graphs. *arXiv preprint arXiv:2504.00993*.
- Jundong Xu, Hao Fei, Liangming Pan, Qian Liu, Mong-Li Lee, and Wynne Hsu. 2024. Faithful logical reasoning via symbolic chain-of-thought. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13326–13365.
- Hao Xue and Flora D Salim. 2023. Promptcast: A new prompt-based learning paradigm for time series forecasting. *IEEE Transactions on Knowledge and Data Engineering*, 36(11):6851–6864.
- An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, and 1 others. 2024. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*.
- Janghoon Yang. 2025. Context information can be more important than reasoning for time series forecasting with a large language model. In *2025 22nd International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)*, pages 1–6. IEEE.
- Shiming Yang, Yuxuan Tong, Xinyao Niu, Graham Neubig, and Xiang Yue. 2025. Demystifying long chain-of-thought reasoning. In *International Conference on Machine Learning*, pages 71177–71209. PMLR.
- Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822.
- Wen Ye, Jinbo Liu, Defu Cao, Wei Yang, and Yan Liu. 2025. When llm meets time series: Can llms perform multistep time series reasoning and inference. In *NeurIPS 2025 Workshop on Evaluating the Evolving LLM Lifecycle: Benchmarks, Emergent Abilities, and Scaling*.
- Yiyao Yu, Yuxiang Zhang, Dongdong Zhang, Xiao Liang, Hengyuan Zhang, Xingxing Zhang, Mahmoud Khademi, Hany Hassan Awadalla, Junjie Wang, Yujia Yang, and 1 others. 2025. Chain-of-reasoning: Towards unified mathematical reasoning in large language models via a multi-paradigm perspective. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 24914–24937.
- Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. 2023. Are transformers effective for time series forecasting? In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pages 11121–11128.
- Haoran Zhang, Yong Liu, Yunzhong Qiu, Haixuan Liu, Zhongyi Pei, Jianmin Wang, and Mingsheng Long. 2025. Timesbert: A bert-style foundation model for time series understanding. In *Proceedings of the 33rd ACM International Conference on Multimedia*, pages 10975–10983.
- Shuyi Zhang, Bin Guo, Anlan Dong, Jing He, Ziping Xu, and Song Xi Chen. 2017. Cautionary tales on air-quality improvement in beijing. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 473(2205).
- Zhuosheng Zhang, Aston Zhang, Mu Li, hai zhao, George Karypis, and Alex Smola. 2024. Multi-modal chain-of-thought reasoning in language models. *Transactions on Machine Learning Research*.
- Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V Le, and Ed H. Chi. 2023a. Least-to-most prompting enables complex reasoning in large language models. In *The Eleventh International Conference on Learning Representations*.
- Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. 2021. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 11106–11115.
- Jingbo Zhou, Xinjiang Lu, Yixiong Xiao, Jiantao Su, Junfu Lyu, Yanjun Ma, and Dejing Dou. 2022. Sdwpf: A dataset for spatial dynamic wind power forecasting challenge at kdd cup 2022. *arXiv preprint arXiv:2208.04360*.
- Tian Zhou, Peisong Niu, Liang Sun, Rong Jin, and 1 others. 2023b. One fits all: Power general time series analysis by pretrained lm. *Advances in neural information processing systems*, 36:43322–43355.
- Zihao Zhou and Rose Yu. 2025. Can llms understand time series anomalies? In *International Conference on Learning Representations*, volume 2025, pages 1858–1896.

A Ethics Statement

This work involves constructing a time-series question answering dataset and evaluating large language models on reasoning tasks. All datasets used are publicly available or synthetically generated for QA purposes, and no personally identifiable information is included. The study does not involve human subjects directly. We note potential ethical considerations in applying LLMs to time-series data in real-world high-stakes domains (e.g., healthcare, finance, energy), where model errors or overconfident reasoning could lead to harmful decisions if used without proper validation, monitoring, and human oversight. Our aim is purely research-focused: to analyze reasoning strategies

and temporal alignment, not to deploy models directly in sensitive decision-making. We encourage users of these findings to implement appropriate safeguards and assess task-specific risks before any practical deployment.

B Broader Impacts

This work provides a systematic analysis of how large language models perform reasoning over time series data across diverse tasks, models, datasets, and reasoning strategies. Its main positive impact lies in clarifying when reasoning strategies are useful and when they may be unreliable for time series analysis. By showing that generic reasoning strategies are not universally beneficial and that their effectiveness depends on temporal decision structures and output spaces, our study can help researchers and practitioners design more appropriate LLM-based tools for domains such as healthcare monitoring, industrial diagnosis, finance, climate analysis, and intelligent transportation.

At the same time, this work also highlights potential risks of applying LLMs to time series tasks without sufficient temporal awareness. In high-stakes scenarios, fluent reasoning explanations may create a false impression of reliability even when the model violates temporal consistency, ignores numerical constraints, or produces overconfident predictions. Such risks are particularly relevant for decision-support settings, where incorrect outputs may lead to inappropriate interventions. Therefore, the methods and findings in this paper should not be interpreted as supporting direct deployment in safety-critical systems without task-specific validation, uncertainty estimation, and human oversight.

Our study is primarily empirical and relies on public datasets together with a self-constructed QA benchmark, which may not fully cover all real-world temporal structures, domain shifts, or user-facing decision environments. The evaluation also involves multiple large language models and repeated sampling, which may introduce non-negligible computational costs. We encourage future work to examine more resource-efficient time-aware reasoning methods, evaluate robustness under real deployment conditions, and develop stronger safeguards for uncertainty calibration, temporal faithfulness, and interpretability.

C Datasets

C.1 Datasets for Classification Task

RCW (Liu et al., 2025b). RCW (North Atlantic Right Whale Calls) is an audio-based time series classification dataset used for detecting right whale calls in oceanic environmental noise. Each sample corresponds to an acoustic signal segment, and the task is to determine whether it contains a North Atlantic right whale call. The key discriminative cue is a short rising “whoop” sound, which provides clear signal-level evidence for the positive class.

TEE (Liu et al., 2025b). TEE (Transient Electromagnetic Event) is a time series classification dataset derived from FORTE satellite observations. The satellite uses optical and radio-frequency instruments to detect transient electromagnetic events associated with lightning activity. Each class corresponds to a specific physical event type with distinguishable temporal characteristics in the measured signal. For example, the “CG Positive” category is related to positive cloud-to-ground discharge and is typically reflected as a sharp radiation spike followed by subsequent signal variations in the power-density time series.

ECG (Liu et al., 2025b). ECG is a single-lead electrocardiogram classification dataset covering four cardiac rhythm categories: normal sinus rhythm, atrial fibrillation, other alternative rhythms, and noisy or unclassifiable recordings. Unlike simple pattern-based tasks, each category depends on multiple clinical signal features. For example, atrial fibrillation is typically identified by combining cues such as irregular rhythm, missing P waves, abnormal baseline patterns, QRS characteristics, and fibrillatory waves.

EMG (Liu et al., 2025b). EMG is a single-channel electromyography classification dataset based on muscle responses to neural stimulation. It includes three categories: healthy, neuropathy, and myopathy. The task requires integrating multiple signal characteristics rather than relying on a single cue. For example, neuropathy is typically associated with fibrillations, positive sharp waves, high-amplitude and long-duration motor unit action potentials, and polyphasic waveforms.

HAR (Liu et al., 2025b). HAR is a human activity recognition dataset collected from smartphone wearable sensors. It records six daily activities, including walking, walking upstairs, walking down-

stairs, sitting, standing, and lying down. We use three sensor channels corresponding to body linear acceleration. Since the data were collected from 30 users with different motion habits, the same activity may exhibit user-specific variations, making HAR suitable for evaluating probabilistic reasoning over behavioral time series.

CTU (Liu et al., 2025b). CTU is a computer type usage dataset based on 24-hour electricity consumption patterns. The task is to infer whether the device is a desktop or a laptop from consumer electricity usage behavior. The dataset was collected from 251 users, whose individual usage habits introduce unobserved variability into the signals.

C.2 Datasets for Anomaly Detection Task

KPI (Li et al., 2022). KPI is a large-scale time series anomaly detection dataset from the 2018 AIOps competition, designed for evaluating anomaly detection algorithms on key performance indicators of online services. It contains both preliminary and final subsets, with approximately six million data points in total. Each sample corresponds to a KPI time series segment, and the task is to identify whether abnormal system behavior occurs at each timestamp or within the sequence. Around 2.5% of the data points are labeled as anomalies, making the dataset highly imbalanced.

Yahoo (Ren et al., 2019). Yahoo is an anomaly detection dataset released by Yahoo Labs for evaluating anomaly detection methods. It contains both synthetic time series and real traffic series collected from Yahoo services. Each sample corresponds to an hourly time series, and the task is to detect abnormal points within the sequence. For the synthetic subset, anomalies are generated algorithmically, while anomalies in the real-traffic subset are manually labeled by human editors. This mixture of simulated and real-world service traffic makes Yahoo suitable for evaluating whether anomaly detection methods can handle both controlled abnormal patterns and realistic operational fluctuations.

WSD (Wang et al., 2024c). WSD is a real-world web service anomaly detection dataset built from key performance indicators collected from large-scale Internet services. It contains KPI time series from three major Internet companies, including Baidu, Sogou, and eBay. Each sample corresponds to a monitored web-service KPI sequence, and the task is to detect anomalous points or abnormal seg-

ments that indicate potential service failures or performance degradation. Since the data are collected from real production systems, WSD reflects practical challenges such as noisy fluctuations, heterogeneous service behaviors, and large-scale operational dynamics, making it suitable for evaluating anomaly detection methods in realistic web-service monitoring scenarios.

NAB (Lavin and Ahmad, 2015). NAB is an open time series anomaly detection benchmark released by Numenta for evaluating streaming anomaly detection algorithms. It contains 58 univariate time series files, where each row includes a timestamp and a scalar value. The data cover diverse domains, including IT metrics, industrial sensors, social media activity, and artificial sequences. The task is to detect anomalous points or intervals under realistic streaming conditions with noise, changing normal patterns, and different anomaly types. NAB contains 365,551 data points in total, and its anomalies are manually labeled through a documented annotation procedure.

NASA-M (Hundman et al., 2018). NASA-M is a real-world time series anomaly detection dataset provided by NASA, derived from telemetry signals of the Curiosity rover. It consists of 27 sensor channels, each represented with 55 feature dimensions. The dataset is split into training and testing sets, where the training data contain only normal patterns and the test data include expert-labeled anomalies. Due to its spacecraft telemetry signals and realistic abnormal events, NASA-M is commonly used to evaluate anomaly detection methods under complex system monitoring scenarios.

NASA-S (Hundman et al., 2018). NASA-S is a real-world time series anomaly detection dataset provided by NASA, derived from spacecraft telemetry signals. It contains 55 telemetry channels with a total of 429,735 data points, including 69 expert-labeled anomalies. Similar to NASA-M, it provides realistic system-monitoring signals with annotated abnormal events, making it commonly used for evaluating anomaly detection methods on multivariate spacecraft telemetry data.

C.3 Datasets for Forecasting Task

ETTh1 (Zhou et al., 2021). ETTh1 is a used public benchmark for long-term time series task, designed to model temperature variations and load dynamics in electric transformers. It contains 2 years

of hourly observations collected from 2016/06 to 2018/06, where each record includes a timestamp, the target variable oil temperature, and six load-related covariates. Following the standard setting, the dataset is split into 12 months for training, 4 months for validation, and 4 months for testing.

ETTh2 (Zhou et al., 2021). ETTh2 is an industrial time series benchmark from the ETT dataset family released by Tsinghua University. It contains hourly observations from 2016 to 2018, with 17,420 samples and seven variables, including six power-load features and oil temperature as the target. Compared with ETTh1, ETTh2 is collected from a transformer station in a different region, where load fluctuations and temperature regulation patterns exhibit distinct regional characteristics.

ETTM1 (Zhou et al., 2021). ETTm1 is a minute-level variant of the ETT benchmark for time series tasks. It contains two years of transformer operation records from July 2016 to July 2018, collected at 15-minute intervals. Each sample includes a timestamp, six power-load covariates, and oil temperature as the target. The load-related variables are HUFL, HULL, MUFL, MULL, LUFL, and LULL, corresponding to high-, middle-, and low-usage load measurements under different operating states. Compared with the hourly ETT datasets, ETTm1 provides finer-grained temporal observations, making it suitable for evaluating models' ability to capture short-term fluctuations.

ETTM2 (Zhou et al., 2021). ETTm2 is minute-level benchmark from the ETT dataset family, designed for fine-grained electric transformer temperature forecasting. It contains two years of records from July 2016 to July 2018, collected at 15-minute intervals. Each record consists of a timestamp, six power-load measurements, and oil temperature as the prediction target. The six covariates, including HUFL, HULL, MUFL, MULL, LUFL, and LULL, describe high-, middle-, and low-usage load behaviors under different operating states.

AQShunyi (Zhang et al., 2017). AQShunyi is an hourly temperature time series dataset collected from the Shunyi air-quality monitoring station. Each sample records temperature observations at regular hourly intervals, and the task is typically to model or analyze temporal patterns in environmental conditions. The dataset exhibits clear seasonal variations, which are consistent with natural temperature cycles across different periods of the year.

Its regular sampling and strong periodic structure make it suitable for evaluating whether time series methods can capture long-term seasonality and environmental dynamics.

NASDAQ (Arashi and Rounaghi, 2022). NASDAQ is a financial time series forecasting dataset composed of daily averaged index data from 2000 to 2016. Each sample records the temporal evolution of stock market index values at a daily resolution, and the task is to predict future index movements based on historical observations. Since financial markets are often affected by regime shifts, volatility changes, and non-stationary dynamics, this dataset is suitable for evaluating whether forecasting models can capture complex temporal dependencies under non-stationary conditions.

NN5 (Teddy and Ng, 2011). NN5 is a daily time series forecasting dataset from the NN5 Forecasting Competition. It consists of withdrawal records collected from 111 randomly selected cash machines located across different regions of England. Each sample corresponds to the daily cash withdrawal sequence of an individual ATM, and the task is to predict future withdrawal demand based on historical observations. Since ATM withdrawals are affected by weekly cycles, holidays, local usage patterns, and irregular demand fluctuations, NN5 is suitable for evaluating whether forecasting models can capture both periodic behavior and non-stationary variations in real-world financial data.

Wind (Zhou et al., 2022). Wind is a large-scale wind power forecasting dataset collected from a wind farm with 134 wind turbines. It contains records sampled every 10 minutes over 245 days, resulting in more than 4.7 million data points. Each sample describes the operational and meteorological status of wind turbines, and the task is to predict future wind power generation from historical observations. Due to the strong variability of wind conditions, turbine-level heterogeneity, and non-stationary power generation patterns, Wind is suitable for evaluating forecasting models in realistic renewable energy scenarios.

C.4 Datasets for Question Answering Task

Overview. To evaluate LLMs' language understanding, semantic alignment, and temporal reasoning abilities in time-series question answering scenarios, we additionally construct a dedicated time-series QA dataset. This QA dataset is inde-

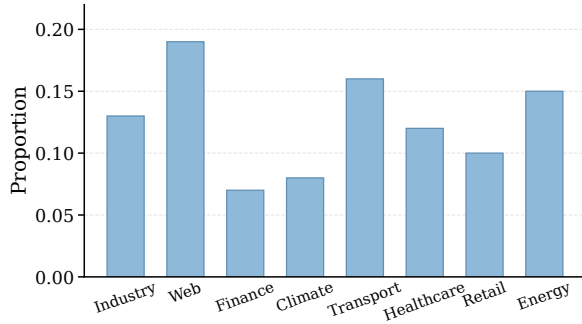


Figure 8: Domain distribution of the QA dataset.

pendent of the datasets used for the preceding classification, anomaly detection, and forecasting tasks, with no sample overlap or task-conversion relationship. It is organized around real-world time-series observations, background information, and question descriptions, requiring models to jointly understand numerical variations, semantic conditions, and task objectives.

Domain Coverage. The question-answering dataset covers eight real-world application domains, including **industry**, **web**, **finance**, **climate**, **transportation**, **healthcare**, **retail**, and **energy**, as shown in Fig. 8. These domains exhibit diverse temporal dynamics. For example, industrial and energy scenarios often involve equipment operating states, electricity demand, power generation, or system stability changes; web-service scenarios focus on access traffic, system load, and abnormal fluctuations; financial scenarios contain market indices, price movements, and trading behaviors; climate scenarios reflect long-term variations in environmental factors such as temperature, wind speed, and precipitation; transportation and retail scenarios capture demand changes, mobility patterns, and consumption behaviors; and healthcare scenarios involve physiological signals or temporal changes in health conditions. Such broad domain coverage ensures that the QA task does not focus on a single type of temporal pattern, but instead spans a wide range of realistic time-series semantic scenarios.

Question Construction. We design four types of question-answering formats: **descriptive**, **relational**, **explanatory**, and **inferential** questions. Descriptive questions require models to identify salient temporal patterns, such as trends, peaks, fluctuations, or periodicity in time series. Relational questions focus on comparisons and dependencies across different time segments, variables, or events. Explanatory questions require models

to connect temporal changes with background information and explain why a certain phenomenon occurs. Inferential questions further ask models to derive possible outcomes or select the most plausible explanation based on available temporal evidence. In total, the QA dataset contains 4,965 true-or-false questions, 5,877 multiple-choice questions, and 6,844 open-ended questions, resulting in approximately 18K question-answer pairs for systematically evaluating LLMs under both closed-form decision making and open-ended reasoning, as shown in Fig. 9.

Pipeline. We adopt a multi-stage pipeline to construct the time-series question-answering dataset. First, we build time-series scenarios by organizing the domain source, variable semantics, temporal range, and background information for each raw time-series sample, thereby forming temporally grounded instances with realistic semantic contexts. Second, we identify question-worthy temporal evidence from each sequence, including trends, peaks and valleys, fluctuation patterns, periodicity, abrupt changes, stage transitions, and inter-variable relations. Only samples with clear and verifiable temporal evidence are retained. Third, we design question types around four categories, namely descriptive, relational, explanatory, and inferential questions, to cover different capability levels from pattern recognition to complex temporal reasoning. Fourth, we generate answers and candidate options for true-or-false, multiple-choice, and open-ended questions. The distractors are constructed based on common reasoning errors rather than random sampling, making the questions more discriminative. Fifth, we conduct quality control by removing samples with insufficient temporal evidence, non-unique answers, ambiguous expressions, or excessive dependence on external knowledge. We further check the consistency of question types, domain distribution, and difficulty levels. Finally, each sample consists of time-series observations, background descriptions, question text, question-type annotations, answers, and candidate options, resulting in an independent time-series question-answering dataset.

Answer Verification and Shortcut Control. To reduce leakage and shortcut risks, we further verify each QA instance after construction. Each question must satisfy three criteria: (1) it is answerable from the given time series and background description; (2) the reference answer is supported by explicit

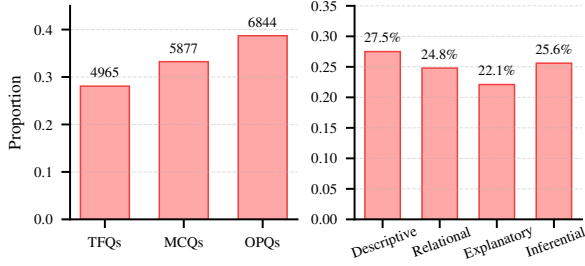


Figure 9: Distribution of QA formats and reasoning types in the constructed time-series question answering dataset.

Table 5: Text-only ablation for evaluating shortcut risks in the QA dataset using DiP.

Setting	TFQs	MCQs	OPQs
Text-only	55.8	30.4	4.7
TS+Text	71.4	65.9	17.2

temporal evidence; and (3) incorrect options or false statements clearly conflict with the temporal evidence. We remove samples with ambiguous answers, insufficient temporal support, multiple plausible interpretations, or questions that can be answered solely from the background description.

We also conduct a text-only ablation to examine whether the benchmark can be solved without accessing the time series. Using DiP, we compare a **Text-only** setting, where the model receives only the background description and question, with a **TS+Text** setting, where the time series is additionally provided. As shown in Tab. 5, removing the time series causes clear performance drops across all QA formats, indicating that the background description alone is insufficient and that temporal evidence is necessary for answering the questions.

D Reasoning Strategy

D.1 General Reasoning Strategies

Direct Prompting (DiP). Direct Prompting is the simplest inference paradigm, where the model directly maps the input time series and task instruction to the final answer without explicitly introducing intermediate reasoning states. Given a time series instance $\mathbf{X}$, a task instruction $\mathcal{I}$, and an output space $\mathcal{Y}$, the LLM parameterized by θ defines a conditional distribution over candidate outputs:

$$p_{\theta}(y \mid \mathbf{X}, \mathcal{I}), \quad y \in \mathcal{Y}.$$

The prediction is obtained by selecting the output with the highest conditional likelihood:

$$\hat{y} = \arg \max_{y \in \mathcal{Y}} p_{\theta}(y \mid \mathbf{X}, \mathcal{I}).$$

For generative outputs, where $y = (y_1, \dots, y_M)$ is a token sequence, the conditional probability is factorized autoregressively as

$$p_{\theta}(y \mid \mathbf{X}, \mathcal{I}) = \prod_{m=1}^M p_{\theta}(y_m \mid y_{<m}, \mathbf{X}, \mathcal{I}).$$

Thus, Direct Prompting performs a one-step implicit decision process, in which task decomposition, temporal evidence extraction, and answer selection are absorbed into the pretrained model distribution. Unlike explicit reasoning strategies, DiP does not require the model to generate or condition on an observable intermediate reasoning trajectory r . Its decision rule can therefore be viewed as a direct prediction:

$$\hat{y} = \arg \max_{y \in \mathcal{Y}} p_{\theta}(y \mid \mathbf{X}, \mathcal{I}),$$

rather than a structured inference process of the form:

$$\hat{y} = \arg \max_{y \in \mathcal{Y}} \sum_{r \in \mathcal{R}} p_{\theta}(y \mid r, \mathbf{X}, \mathcal{I}) p_{\theta}(r \mid \mathbf{X}, \mathcal{I}),$$

where $\mathcal{R}$ denotes the space of explicit reasoning trajectories. This makes DiP computationally efficient, but also limits its ability to enforce grounded intermediate decisions, especially for tasks requiring multi-step comparison, temporal dependency modeling, or uncertainty-aware reasoning.

Chain-of-Thought (CoT). Chain-of-Thought introduces an explicit intermediate reasoning trajectory before generating the final answer. Given a time series instance $\mathbf{X}$, a task instruction $\mathcal{I}$, and an output space $\mathcal{Y}$, CoT augments the direct prediction process with a sequence of latent-but-observable reasoning states $r = (r_1, \dots, r_T)$, where each r_t corresponds to one intermediate reasoning step. The model therefore defines a joint distribution over the reasoning trajectory and the final output:

$$p_{\theta}(y, r \mid \mathbf{X}, \mathcal{I}) = p_{\theta}(r \mid \mathbf{X}, \mathcal{I}) p_{\theta}(y \mid r, \mathbf{X}, \mathcal{I}).$$

The reasoning trajectory is generated autoregressively as:

$$p_{\theta}(r \mid \mathbf{X}, \mathcal{I}) = \prod_{t=1}^T p_{\theta}(r_t \mid r_{<t}, \mathbf{X}, \mathcal{I}),$$

which yields the complete factorization:

$$p_{\theta}(y, r \mid \mathbf{X}, \mathcal{I}) = \left[\prod_{t=1}^T p_{\theta}(r_t \mid r_{<t}, \mathbf{X}, \mathcal{I}) \right] \cdot p_{\theta}(y \mid r, \mathbf{X}, \mathcal{I}).$$

The final prediction can be viewed as selecting the output supported by the generated reasoning trajectory:

$$\hat{y} = \arg \max_{y \in \mathcal{Y}} p_{\theta}(y \mid \hat{r}, \mathbf{X}, \mathcal{I}), \quad \hat{r} \sim p_{\theta}(r \mid \mathbf{X}, \mathcal{I}).$$

Compared with Direct Prompting, CoT externalizes part of the inference process into a linear reasoning path, allowing the model to decompose the task into intermediate steps such as temporal evidence identification, comparison, and answer justification. However, because the final decision is conditioned on a single generated trajectory, errors in early reasoning steps may be propagated or amplified in later steps, especially when the generated chain is not faithful to the underlying time series.

Least-to-Most (L2M). Least-to-Most reasoning decomposes a complex task into an ordered sequence of simpler sub-problems and solves them progressively. Given a time series instance $\mathbf{X}$ and a task instruction $\mathcal{I}$, L2M first generates a decomposition $q = (q_1, \dots, q_K)$, where each q_k denotes a sub-problem with lower reasoning complexity:

$$q \sim p_{\theta}(q \mid \mathbf{X}, \mathcal{I}).$$

The decomposition can be factorized as:

$$p_{\theta}(q \mid \mathbf{X}, \mathcal{I}) = \prod_{k=1}^K p_{\theta}(q_k \mid q_{<k}, \mathbf{X}, \mathcal{I}).$$

After obtaining the ordered sub-problems, the model solves them sequentially. Let a_k denote the intermediate answer to the k -th sub-problem. Each intermediate answer is conditioned on the current sub-problem and all previous sub-problem answers:

$$p_{\theta}(a \mid q, \mathbf{X}, \mathcal{I}) = \prod_{k=1}^K p_{\theta}(a_k \mid q_k, a_{<k}, q_{<k}, \mathbf{X}, \mathcal{I}),$$

where $a = (a_1, \dots, a_K)$ denotes the sequence of intermediate answers. The final prediction is then generated by aggregating the solved sub-problems:

$$\hat{y} = \arg \max_{y \in \mathcal{Y}} p_{\theta}(y \mid a, q, \mathbf{X}, \mathcal{I}).$$

Equivalently, L2M can be viewed as structured inference over both the decomposition path q and the intermediate answer path a :

$$\begin{aligned} p_{\theta}(y, a, q \mid \mathbf{X}, \mathcal{I}) &= p_{\theta}(q \mid \mathbf{X}, \mathcal{I}) \\ &\cdot p_{\theta}(a \mid q, \mathbf{X}, \mathcal{I}) \\ &\cdot p_{\theta}(y \mid a, q, \mathbf{X}, \mathcal{I}). \end{aligned}$$

Compared with CoT, which follows a single linear reasoning trajectory, L2M imposes an explicit problem-decomposition structure. This makes it suitable for tasks where the final decision can be built from simpler temporal judgments, such as identifying local patterns before comparing them globally. However, if the decomposition does not align with the true temporal decision structure, errors in early sub-problems may constrain later reasoning and lead to suboptimal predictions.

Tree-of-Thoughts (ToT). Tree-of-Thoughts extends linear reasoning by formulating inference as a tree-structured search over multiple candidate reasoning trajectories. Given a time series instance $\mathbf{X}$ and a task instruction $\mathcal{I}$, each node s_t represents a partial reasoning state at depth t , and each root-to-node path corresponds to a partial reasoning trajectory. Starting from the initial state s_0 , the model expands a set of candidate successor states:

$$\begin{aligned} \mathcal{C}(s_t) &= \{s_{t+1}^{(1)}, \dots, s_{t+1}^{(B)}\}, \\ s_{t+1}^{(b)} &\sim p_{\theta}(s_{t+1} \mid s_t, \mathbf{X}, \mathcal{I}), \end{aligned}$$

where B denotes the branching factor. Each candidate state is evaluated by a value function: $V_{\theta}(s_t; \mathbf{X}, \mathcal{I}) \in \mathbb{R}$, which estimates the promise of the partial reasoning state for producing a correct final answer. At each search step, ToT selects a subset of promising states from the expanded candidates. Let $\mathcal{S}_t$ denote the set of active states at depth t . The next active set is obtained by

$$\mathcal{S}_{t+1} = \text{TopK}_{s \in \bigcup_{\bar{s} \in \mathcal{S}_t} \mathcal{C}(\bar{s})} V_{\theta}(s; \mathbf{X}, \mathcal{I}),$$

where $\text{TopK}(\cdot)$ keeps the K highest-scoring partial reasoning states. After reaching the maximum depth T or a terminal condition, the final answer is selected from the highest-scoring complete path:

$$\begin{aligned} \hat{s} &= \arg \max_{s \in \mathcal{S}_T} V_{\theta}(s; \mathbf{X}, \mathcal{I}), \\ \hat{y} &= \arg \max_{y \in \mathcal{Y}} p_{\theta}(y \mid \hat{s}, \mathbf{X}, \mathcal{I}). \end{aligned}$$

Compared with CoT and L2M, ToT maintains multiple candidate trajectories and delays commitment until after intermediate evaluation. This search-based formulation can reduce early reasoning errors and improve robustness under ambiguous temporal evidence. However, its effectiveness depends on whether the value function can reliably evaluate partial temporal reasoning states; otherwise, the search may prioritize fluent but temporally inconsistent branches.

Self-Refine (SRF). Self-Refine improves an initial prediction through iterative self-feedback and revision. Given a time series instance $\mathbf{X}$, a task instruction $\mathcal{I}$, and an output space $\mathcal{Y}$, the model first generates an initial answer:

$$y^{(0)} \sim p_\theta(y \mid \mathbf{X}, \mathcal{I}).$$

At the i -th refinement step, the model produces a critique or feedback signal $c^{(i)}$ conditioned on the input and the current answer:

$$c^{(i)} \sim p_\theta(c \mid y^{(i)}, \mathbf{X}, \mathcal{I}).$$

The current answer is then updated according to the generated feedback:

$$y^{(i+1)} \sim p_\theta(y \mid y^{(i)}, c^{(i)}, \mathbf{X}, \mathcal{I}).$$

After R refinement rounds, the final prediction is obtained as:

$$\hat{y} = y^{(R)}.$$

Equivalently, the iterative refinement process induces the following trajectory distribution:

$$p_\theta(y^{(0:R)}, c^{(0:R-1)} \mid \mathbf{X}, \mathcal{I}) = p_\theta(y^{(0)} \mid \mathbf{X}, \mathcal{I}) \cdot \prod_{i=0}^{R-1} p_\theta(c^{(i)} \mid y^{(i)}, \mathbf{X}, \mathcal{I}) \cdot p_\theta(y^{(i+1)} \mid y^{(i)}, c^{(i)}, \mathbf{X}, \mathcal{I}).$$

SRF repeatedly revisits the current prediction and attempts to correct it using self-generated feedback. This formulation can improve robustness when the initial answer contains detectable inconsistencies, such as incorrect temporal comparisons or unsupported explanations. However, because both the critique and refinement are generated by the same model, SRF may also reinforce erroneous assumptions when the feedback signal fails to identify violations of temporal evidence.

D.2 Time-Aware Reasoning Strategies

Temporal Regime Segmentation (TRS). Many real-world time series exhibit heterogeneous temporal regimes, where the underlying dynamics vary across different stages. Directly reasoning over the entire sequence may mix incompatible temporal patterns and obscure stage-specific evidence. Temporal Regime Segmentation addresses this issue by decomposing the input sequence into locally coherent regimes and performing reasoning at the segment level before making the final decision.

Given a time series instance $\mathbf{X} = \{\mathbf{x}_t\}_{t=1}^T$, TRS assumes a latent segmentation induced by change points $\tau = (\tau_0, \tau_1, \dots, \tau_K)$, where $0 = \tau_0 < \tau_1 < \dots < \tau_K = T$. The sequence is partitioned as

$$\mathbf{X} = \bigcup_{k=1}^K \mathbf{X}^{(k)}, \quad \mathbf{X}^{(k)} = \{\mathbf{x}_t\}_{t=\tau_{k-1}+1}^{\tau_k}.$$

Each segment $\mathbf{X}^{(k)}$ corresponds to a local temporal regime with relatively coherent dynamics. The segmentation process can be viewed as estimating the most plausible regime partition:

$$\hat{\tau} = \arg \max_{\tau} p_\theta(\tau \mid \mathbf{X}, \mathcal{I}).$$

After segmentation, the model performs regime-level reasoning over each segment. Let r_k denote the reasoning outcome or temporal evidence extracted from the k -th regime:

$$r_k \sim p_\theta(r_k \mid \mathbf{X}^{(k)}, \mathcal{I}), \quad k = 1, \dots, K.$$

The collection of regime-level reasoning outcomes is denoted as $r_{1:K} = (r_1, \dots, r_K)$. The final prediction is then generated by aggregating these segment-level results:

$$\hat{y} = \arg \max_{y \in \mathcal{Y}} p_\theta(y \mid r_{1:K}, \mathbf{X}, \mathcal{I}).$$

Equivalently, TRS can be formulated as structured inference over both latent regime boundaries and regime-specific reasoning states:

$$p_\theta(y, r_{1:K}, \tau \mid \mathbf{X}, \mathcal{I}) = p_\theta(\tau \mid \mathbf{X}, \mathcal{I}) \cdot \prod_{k=1}^K p_\theta(r_k \mid \mathbf{X}^{(k)}, \mathcal{I}) \cdot p_\theta(y \mid r_{1:K}, \mathbf{X}, \mathcal{I}).$$

This formulation separates local regime understanding from global decision making. By first identifying stage-specific temporal evidence and then integrating it across regimes, TRS reduces interference between heterogeneous temporal patterns and provides a more temporally aligned reasoning structure, especially for tasks involving regime shifts, phase transitions, or localized abnormal behaviors.

Latent Dynamics Modeling (LDM). While TRS reasons over explicit temporal regimes, many time-dependent processes evolve through continuous and partially unobserved system states. Directly reasoning over raw observations may be

sensitive to local noise, irregular fluctuations, or short-term perturbations. Latent Dynamics Modeling addresses this issue by introducing a latent trajectory that compactly summarizes the underlying temporal evolution and supports reasoning at the state-dynamics level.

Given a time series instance $\mathbf{X} = \{\mathbf{x}_t\}_{t=1}^T$, LDM assumes that each observation is generated from an unobserved latent state $\mathbf{z}_t$. The latent state evolves according to a transition process:

$$\mathbf{z}_t \sim p_\theta(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{u}_t, \mathcal{I}), \quad \mathbf{x}_t \sim p_\theta(\mathbf{x}_t \mid \mathbf{z}_t),$$

where $\mathbf{z}_t$ denotes the latent system state at time t , $\mathbf{u}_t$ represents exogenous or contextual information, and $\mathcal{I}$ denotes the task instruction. The joint distribution over the observed sequence and latent trajectory can be written as

$$p_\theta(\mathbf{X}, \mathbf{Z} \mid \mathbf{U}, \mathcal{I}) = \prod_{t=1}^T p_\theta(\mathbf{x}_t \mid \mathbf{z}_t) p_\theta(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{u}_t, \mathcal{I}),$$

where $\mathbf{Z} = \{\mathbf{z}_t\}_{t=1}^T$ and $\mathbf{U} = \{\mathbf{u}_t\}_{t=1}^T$.

Instead of making decisions from individual observations, LDM first infers a latent trajectory that explains the temporal evolution:

$$\hat{\mathbf{Z}} = \arg \max_{\mathbf{Z}} p_\theta(\mathbf{Z} \mid \mathbf{X}, \mathbf{U}, \mathcal{I}).$$

The final prediction is then generated by conditioning on the inferred latent trajectory:

$$\hat{y} = \arg \max_{y \in \mathcal{Y}} p_\theta(y \mid \hat{\mathbf{Z}}, \mathbf{X}, \mathcal{I}).$$

Equivalently, LDM can be viewed as marginalizing over possible latent temporal trajectories:

$$p_\theta(y \mid \mathbf{X}, \mathbf{U}, \mathcal{I}) = \int p_\theta(y \mid \mathbf{Z}, \mathbf{X}, \mathcal{I}) p_\theta(\mathbf{Z} \mid \mathbf{X}, \mathbf{U}, \mathcal{I}) d\mathbf{Z}.$$

This formulation encourages the model to reason over the underlying temporal dynamics rather than isolated observed values. By abstracting raw observations into latent states, LDM can suppress local noise, preserve long-range dependencies, and provide a more stable reasoning structure for forecasting, explanatory QA, and tasks requiring integration of distributed temporal evidence.

Distributional Output Modeling (DOM). Time-aware reasoning often operates under substantial uncertainty caused by observation noise, non-stationarity, sparse temporal evidence, and ambiguous semantic conditions. A deterministic output

may be insufficient, since it only returns a single answer without characterizing the reliability of the decision. Distributional Output Modeling addresses this issue by representing the reasoning result as a conditional distribution over possible outputs.

Given a time series instance $\mathbf{X}$, a task instruction $\mathcal{I}$, and an output space $\mathcal{Y}$, DOM models the predictive distribution as: $p_\theta(y \mid \mathbf{X}, \mathcal{I})$, $y \in \mathcal{Y}$. In practice, this distribution can be approximated by drawing multiple candidate outputs:

$$y^{(n)} \sim p_\theta(y \mid \mathbf{X}, \mathcal{I}), \quad n = 1, \dots, N,$$

which yields an empirical output distribution

$$\hat{p}(y \mid \mathbf{X}, \mathcal{I}) = \frac{1}{N} \sum_{n=1}^N \mathbb{I}[y^{(n)} = y],$$

for discrete output spaces. For open-ended outputs, the same idea can be implemented by clustering or scoring semantically equivalent responses before estimating their empirical probabilities.

The uncertainty of each candidate output is then computed from the predictive distribution:

$$\text{Unc}(y) = \mathcal{U}(\hat{p}(y \mid \mathbf{X}, \mathcal{I})),$$

where $\mathcal{U}(\cdot)$ denotes an uncertainty functional. Instead of selecting the most frequent output alone, DOM chooses the answer by jointly considering task-specific utility and predictive uncertainty:

$$\hat{y} = \arg \max_{y \in \mathcal{Y}} [\mathbb{E}_{\hat{p}}[U(y; \mathbf{X}, \mathcal{I})] - \lambda \text{Unc}(y)],$$

where $U(\cdot)$ denotes the task-specific utility function and λ controls the strength of uncertainty penalization. DOM preserves multiple plausible reasoning outcomes and reduces overconfident decisions under ambiguous temporal evidence. It is particularly useful for tasks where the correct output depends on noisy observations, weak temporal cues, or multiple competing interpretations.

E Baselines

E.1 Specialized Time Series Models

TFPS (Sun et al., 2026). TFPS is a pattern-specific expert model designed for time series forecasting. Its key idea is to assign different forecasting experts to heterogeneous patterns appearing in different temporal segments. Real-world time series often contain diverse dynamics, such as seasonality, regime transitions, and contextual

drift. A single unified model may therefore suffer from patch-level distribution shifts when fitting all temporal segments with the same forecasting function. By introducing pattern-specific experts, TFPS selects suitable prediction functions according to local temporal patterns, thereby improving its ability to model complex pattern evolution and distributional changes. We include TFPS as a specialized lightweight time-series baseline to examine the difference between generic LLM-based reasoning strategies and dedicated temporal architectures.

TSLANet (Eldele et al., 2024). TSLANet is a lightweight time series representation learning model designed as an efficient alternative to Transformer-based models, which often involve large parameter sizes, high computational costs, and sensitivity to noise. Its core design combines frequency-domain analysis with convolutional modeling. Specifically, the Adaptive Spectral Block captures long-range dependencies and periodic structures, while convolutional modules model local short-term patterns. Compared with standard Transformers, TSLANet places greater emphasis on computational efficiency, noise robustness, and stable performance on relatively small datasets. We therefore use it as a non-LLM and non-foundation-model baseline specialized for time series modeling.

TimeXer (Wang et al., 2024b). TimeXer is a Transformer-based forecasting model for time series with exogenous variables. Unlike conventional multivariate forecasting methods that treat all variables uniformly, TimeXer explicitly distinguishes endogenous variables from exogenous variables. It integrates external information through specialized embeddings, patch-wise self-attention, and variate-wise cross-attention. The goal is to use exogenous variables as additional contextual signals for improving the prediction of target variables. Since many real-world time series depend not only on their own historical dynamics but also on external factors, TimeXer serves as a specialized lightweight baseline for modeling exogenous information and cross-variable dependencies.

E.2 LLM for Time Series

LLM-TS Integrator (Chen et al., 2025). LLM-TS Integrator is a framework that incorporates the capabilities of LLMs into conventional time series modeling. Unlike methods that directly use an LLM as the forecasting backbone, it preserves

the strengths of traditional time series models in capturing periodicity, local patterns, and numerical variations, while leveraging the high-level semantic and pattern-recognition information provided by LLMs to enhance the forecasting process. Specifically, LLM-TS Integrator employs a time-series branch to extract numerical features, and uses a mutual information module to interact with and constrain the representations from the LLM side. In this way, the time series model can benefit from the prior knowledge of LLMs without fully relying on language models for numerical prediction.

TimeCMA (Liu et al., 2025a). TimeCMA is an LLM-empowered framework for multivariate time series forecasting. Its core idea is to mitigate the modality gap between time-series representations and the semantic space of language models through cross-modality alignment. The framework contains two encoding branches: a time-series encoding branch that extracts relatively decoupled temporal embeddings, and an LLM-empowered branch that combines time series with textual prompts to obtain more robust semantic representations. By aligning the two modalities, TimeCMA aims to exploit both local temporal patterns and the contextual modeling capability of LLMs.

SE-LLM (Liu et al., 2026b). SE-LLM is a semantic-enhanced time-series LLM method that aims to alleviate the modality gap between time series and language models at the semantic level. Unlike methods that rely only on token-level alignment, SE-LLM explicitly exploits periodic and anomalous patterns in time series to enhance token embeddings, so that time-series tokens carry clearer temporal-pattern semantics in the LLM embedding space. It also introduces plug-in modules tailored for time series forecasting, which strengthen the LLM’s ability to model both long- and short-term temporal dependencies as well as local abnormal variations.

FSCA (Hu et al., 2025). FSCA is derived from the Context-Alignment framework. Its central idea is to construct time-series context structures that are more compatible with LLM understanding, rather than relying solely on token-level alignment. It uses Dual-Scale Context-Alignment GNNs to model structural and logical relations in time series, and further combines them with few-shot prompting to form Few-Shot prompting Context-Alignment, namely FSCA. FSCA can be flexibly

integrated into different layers of a pretrained LLM to enhance the model’s awareness of temporal structure, contextual logic, and task semantics.

F Implementation Details

F.1 Unified Inference Pipeline

To ensure fair comparison and reproducibility, all reasoning strategies follow the same inference pipeline. Given the raw time series, timestamps, optional background description, task instruction, and task-specific output space, we serialize the input into a unified textual format and insert it into a fixed prompt template. For the same task, all methods use identical input representation, window length, prediction horizon, decoding parameters, and sampling budget, without additional normalization, scaling, truncation, or adaptive reformatting unless explicitly specified in preprocessing ablations. The pipeline includes input construction, strategy-specific reasoning organization, LLM inference, output parsing, and result aggregation. Specifically, TRS generates continuous temporal segments covering the full index range as intermediate reasoning structures; LDM produces a structured latent trajectory of dynamic states before prediction; and DOM samples multiple candidate outputs to form an empirical distribution for uncertainty-aware decisions. A unified parser extracts labels, options, anomaly judgments, predicted sequences, or open-ended answers from model outputs. For discrete tasks, valid candidates are aggregated by majority voting, with invalid outputs handled by predefined parsing rules. For point- or interval-level anomaly detection, sampled anomaly markers are mapped to a unified time index, voted position-wise, and merged into final anomaly intervals. For forecasting, invalid or length-mismatched sequences are filtered out, and the remaining predictions are averaged element-wise across future time steps.

F.2 Implementation Details of TRS

The core idea of TRS is to explicitly divide the complete time series into several continuous temporal segments, enabling the model to first reason over local temporal states and then aggregate segment-level reasoning results into a global prediction. It is important to note that TRS does not rely on any external change-point detection algorithm, nor does it modify the original numerical values according to the segmentation results. All segmentation boundaries are proposed by the LLM during inference

based on the temporal variation patterns of the input sequence, and are used only as intermediate reasoning structures.

1. **Input preparation.** Given a raw time series $\mathbf{x}_{1:T}$, timestamps, optional background description, and task instruction, we serialize them into a unified textual prompt.
2. **Temporal segmentation proposal.** The LLM is instructed to propose at most K continuous temporal segments according to the temporal variation patterns observed from the input. The generated segments are required to cover the complete index range without gaps.
3. **Local segment reasoning.** For each proposed segment, the LLM analyzes the local temporal state, such as stability, increasing or decreasing trend, abrupt change, periodic fluctuation, or abnormal deviation.
4. **Segment-level evidence organization.** The model summarizes the evidence from all segments and identifies how different local temporal states contribute to the target task.
5. **Global prediction.** Based on the organized segment-level reasoning, the model produces the final task-specific output, such as a class label, option choice, anomaly judgment, forecast sequence, or open-ended answer.
6. **Output parsing.** A unified parser extracts the final answer from the model output. The intermediate segment boundaries are only used for reasoning interpretation and are not used to modify, normalize, truncate, or rescale the original time series.

TRS Classification Prompt Template

You are an expert in time series analysis and temporal reasoning. Given a time series, your task is to classify it into one of the candidate categories. You must reason based only on the provided time series, timestamps, background information, and task instruction.

Input Time Series: {TIME_SERIES}

Timestamps: {TIMESTAMPS}

Background Information: {BACKGROUND}

Task Instruction: {TASK_INSTRUCTION}

Candidate Labels: {LABEL_SPACE}

Please follow the TRS reasoning procedure:

1. Divide the complete time series into at most {K} continuous temporal segments. The segments must cover the full index range from the first time step to the last time step without overlap or gaps.
2. For each segment, describe its local temporal pattern, such as stable behavior, increasing trend, decreasing trend, fluctuation, abrupt change, periodic pattern, or abnormal deviation.
3. Summarize how the segment-level temporal patterns support the classification decision.
4. Output exactly one final label from the candidate labels.

Please use the following output format:

Segments: - Segment 1: [start index, end index], local pattern: ... - Segment 2: [start index, end index], local pattern:

Global Reasoning: ...

Final Answer: <label>

Figure 10: Prompt template for classification tasks with TRS.

TRS Anomaly Detection Prompt Template

You are an expert in time series analysis and anomaly detection. Given a time series, your task is to determine whether anomalous behavior exists and, if required, identify the anomalous time points or intervals. You must reason based only on the provided time series, timestamps, background information, and task instruction.

Input Time Series: {TIME_SERIES}

Timestamps: {TIMESTAMPS}

Background Information: {BACKGROUND}

Task Instruction: {TASK_INSTRUCTION}

Output Space: {OUTPUT_SPACE}

Please follow the TRS reasoning procedure:

1. Divide the complete time series into at most {K} continuous temporal segments. The segments must cover the full index range from the first time step to the last time step without overlap or gaps.
2. For each segment, describe its local temporal behavior, such as stable behavior, increasing trend, decreasing trend, fluctuation, abrupt change, level shift, spike, drop, or abnormal deviation.
3. Compare the segment-level patterns with the expected temporal behavior described in the task and background information.
4. Determine whether the time series contains anomalies. If the task requires point-level or interval-level detection, specify the anomalous time points or continuous anomalous intervals.

Please use the following output format:

Segments: - Segment 1: [start index, end index], local pattern: ... - Segment 2: [start index, end index], local pattern:

Anomaly Evidence: ...

Final Answer: <normal / anomalous / anomaly points / anomaly intervals>

Figure 11: Prompt template for anomaly detection tasks with TRS.

TRS Forecasting Prompt

You are an expert in time series analysis and forecasting. Given a historical time series, your task is to predict its future values for the specified forecasting horizon. You must reason based only on the provided time series, timestamps, background information, and task instruction.

Input Time Series: {TIME_SERIES}

Timestamps: {TIMESTAMPS}

Background Information: {BACKGROUND}

Task Instruction: {TASK_INSTRUCTION}

Forecasting Horizon: {PREDICTION_LENGTH}

Please follow the TRS reasoning procedure:

1. Divide the historical time series into at most {K} continuous temporal segments. The segments must cover the full index range from the first historical time step to the last historical time step without overlap or gaps.
2. For each segment, describe its local temporal behavior, such as stable behavior, increasing trend, decreasing trend, fluctuation, periodic pattern, level shift, abrupt change, or abnormal deviation.
3. Summarize how the segment-level temporal patterns indicate the future dynamics, including trend continuation, reversal, periodic recurrence, stabilization, or uncertainty.
4. Generate a forecast sequence with exactly {PREDICTION_LENGTH} numerical values.

Please use the following output format:

Segments: - Segment 1: [start index, end index], local pattern: ... - Segment 2: [start index, end index], local pattern:

Forecasting Evidence: ...

Final Answer: <v_1, v_2, ..., v_{PREDICTION_LENGTH}>

Figure 12: Prompt template for forecasting tasks with TRS.

TRS Prompt Template for TFQs

You are an expert in time series analysis and temporal reasoning. Given a time series and a true-or-false question, your task is to determine whether the statement is true or false. You must reason based only on the provided time series, timestamps, background information, and task instruction.

Input Time Series: {TIME_SERIES}

Timestamps: {TIMESTAMPS}

Background Information: {BACKGROUND}

Question: {QUESTION}

Candidate Answers: True, False

Please follow the TRS reasoning procedure:

1. Divide the complete time series into at most {K} continuous temporal segments. The segments must cover the full index range without overlap or gaps.
2. For each segment, describe its local temporal pattern, such as stable behavior, increasing trend, decreasing trend, fluctuation, abrupt change, periodic pattern, or abnormal deviation.
3. Compare the segment-level evidence with the statement in the question.
4. Output exactly one final answer: True or False.

Please use the following output format:

Segments: - Segment 1: [start index, end index], local pattern: ... - Segment 2: [start index, end index], local pattern:

Evidence: ...

Final Answer: <True / False>

Figure 13: Prompt template for true-or-false questions with TRS.

TRS Prompt Template for MCQs

You are an expert in time series analysis and temporal reasoning. Given a time series and a multiple-choice question, your task is to select the correct option. You must reason based only on the provided time series, timestamps, background information, question, and candidate options.

Input Time Series: {TIME_SERIES}

Timestamps: {TIMESTAMPS}

Background Information: {BACKGROUND}

Question: {QUESTION}

Candidate Options: {OPTIONS}

Please follow the TRS reasoning procedure:

1. Divide the complete time series into at most {K} continuous temporal segments. The segments must cover the full index range without overlap or gaps.
2. For each segment, describe its local temporal pattern, such as stable behavior, increasing trend, decreasing trend, fluctuation, abrupt change, periodic pattern, or abnormal deviation.
3. Compare the segment-level temporal evidence with each candidate option.
4. Output exactly one final option from the candidate options.

Please use the following output format:

Segments: - Segment 1: [start index, end index], local pattern: ... - Segment 2: [start index, end index], local pattern:

Option Analysis: ...

Final Answer: <option label>

Figure 14: Prompt template for multiple-choice questions with TRS.

TRS Prompt Template for OPQs

You are an expert in time series analysis and temporal reasoning. Given a time series and an open-ended question, your task is to generate a concise answer based on the temporal evidence. You must reason based only on the provided time series, timestamps, background information, and question.

Input Time Series: {TIME_SERIES}

Timestamps: {TIMESTAMPS}

Background Information: {BACKGROUND}

Question: {QUESTION}

Please follow the TRS reasoning procedure:

1. Divide the complete time series into at most {K} continuous temporal segments. The segments must cover the full index range without overlap or gaps.
2. For each segment, describe its local temporal pattern, such as stable behavior, increasing trend, decreasing trend, fluctuation, abrupt change, periodic pattern, or abnormal deviation.
3. Summarize the segment-level evidence relevant to the question.
4. Provide a concise final answer grounded in the observed temporal patterns.

Please use the following output format:

Segments: - Segment 1: [start index, end index], local pattern: ... - Segment 2: [start index, end index], local pattern:

Evidence: ...

Final Answer: <concise textual answer>

Figure 15: Prompt template for open-ended questions with TRS.

F.3 Implementation Details of LDM

The core idea of LDM is to represent the temporal evolution of a time series through a sequence of structured latent dynamic states, enabling the model to reason over state transitions rather than isolated observations. Different from external state-space modeling or learned latent-variable models, LDM does not introduce additional trainable modules or externally computed latent variables. Instead, the latent states are represented as structured textual summaries generated by the LLM during inference and are used as intermediate reasoning structures before producing the final task-specific prediction. LDM follows a structured reasoning procedure that explicitly separates latent trajectory construction from final decision making. Its implementation can be summarized as follows:

1. **Input preparation.** Given a raw time series $x_{1:T}$, timestamps, optional background description, and task instruction, we serialize them into the same unified textual format used by other reasoning strategies.
2. **Latent state abstraction.** The LLM is instructed to summarize the temporal evolution of the input sequence into a small number of ordered latent dynamic states. Each state describes a dominant temporal condition, such as stable fluctuation, upward drift, downward drift, abrupt transition, periodic oscillation, or increasing volatility.
3. **Latent trajectory construction.** The generated latent states are connected sequentially to form a latent trajectory. This trajectory provides a compact description of how the underlying temporal dynamics evolve over time, for example from a stable regime to a transition state and then to an upward drifting state.
4. **State-level reasoning.** For each latent state, the LLM analyzes its relevance to the target task. For classification and question answering, the model identifies which states provide discriminative evidence. For anomaly detection, the model checks whether a state deviates from the normal dynamic pattern. For forecasting, the model uses the recent latent states to infer the future direction and magnitude of change.
5. **Trajectory-level aggregation.** The model aggregates evidence across the latent trajectory

by considering both individual state properties and transitions between states. In particular, LDM emphasizes temporal dependencies such as persistence, reversal, acceleration, transition, and regime shift, rather than treating each observation or segment independently.

6. **Global prediction.** Based on the latent trajectory and the aggregated state-level evidence, the model produces the final task-specific output, such as a class label, option choice, anomaly judgment, forecast sequence, or open-ended answer.
7. **Output parsing.** A unified parser extracts the final answer from the model output. The latent states are used only as intermediate textual reasoning structures and are not treated as externally computed variables or additional model inputs.

LDM Classification Prompt Template

You are an expert in time series analysis and temporal reasoning. Given a time series, your task is to classify it into one of the candidate categories. You must reason based only on the provided time series, timestamps, background information, and task instruction.

Input Time Series: {TIME_SERIES}

Timestamps: {TIMESTAMPS}

Background Information: {BACKGROUND}

Task Instruction: {TASK_INSTRUCTION}

Candidate Labels: {LABEL_SPACE}

Please follow the LDM reasoning procedure:

1. First describe the latent dynamic trajectory of the time series as a sequence of dynamic states, such as stable state, upward drift, downward drift, turning point, periodic fluctuation, intensified fluctuation, level shift, or abnormal deviation.
2. Explain how these latent dynamic states evolve over time and how they are related to the target classification task.
3. Use the latent dynamic trajectory as the main evidence to determine the final class.
4. Output exactly one final label from the candidate labels.

Please use the following output format:

Latent Dynamic Trajectory: ...

Classification Evidence: ...

Final Answer: <label>

Figure 16: Prompt template for classification tasks with LDM.

LDM Anomaly Detection Prompt Template

You are an expert in time series analysis and anomaly detection. Given a time series, your task is to determine whether anomalous behavior exists and, if required, identify the anomalous time points or intervals. You must reason based only on the provided time series, timestamps, background information, and task instruction.

Input Time Series: {TIME_SERIES}

Timestamps: {TIMESTAMPS}

Background Information: {BACKGROUND}

Task Instruction: {TASK_INSTRUCTION}

Output Space: {OUTPUT_SPACE}

Please follow the LDM reasoning procedure:

1. First describe the latent dynamic trajectory of the time series as a sequence of dynamic states, such as normal stability, gradual drift, fluctuation increase, abrupt spike, sudden drop, level shift, turning point, or abnormal deviation.
2. Identify whether any dynamic state deviates from the expected temporal behavior described by the task and background information.
3. If anomalies exist, determine whether they correspond to sequence-level, point-level, or interval-level abnormal behavior.
4. Output the final anomaly judgment or the anomalous time points/intervals according to the task requirement.

Please use the following output format:

Latent Dynamic Trajectory: ...

Anomaly Evidence: ...

Final Answer: <normal / anomalous / anomaly points / anomaly intervals>

Figure 17: Prompt template for anomaly detection tasks with LDM.

LDM Forecasting Prompt Template

You are an expert in time series analysis and forecasting. Given a historical time series, your task is to predict its future values for the specified forecasting horizon. You must reason based only on the provided time series, timestamps, background information, and task instruction.

Input Time Series: {TIME_SERIES}

Timestamps: {TIMESTAMPS}

Background Information: {BACKGROUND}

Task Instruction: {TASK_INSTRUCTION}

Forecasting Horizon: {PREDICTION_LENGTH}

Please follow the LDM reasoning procedure:

1. First infer the latent dynamic trajectory of the historical time series, including dynamic states such as stability, upward drift, downward drift, periodic fluctuation, trend acceleration, trend reversal, level shift, or intensified volatility.
2. Explain how the inferred latent trajectory is likely to evolve into the future, considering trend continuation, reversal, periodic recurrence, stabilization, or uncertainty.
3. Generate a forecast sequence based on the latent dynamic trajectory.
4. Output exactly {PREDICTION_LENGTH} numerical values.

Please use the following output format:

Latent Dynamic Trajectory: ...

Forecasting Evidence: ...

Final Answer: <v_1, v_2, ..., v_{PREDICTION_LENGTH}>

Figure 18: Prompt template for forecasting tasks with LDM.

LDM Prompt Template for TFQs

You are an expert in time series analysis and temporal reasoning. Given a time series and a true-or-false question, your task is to determine whether the statement is true or false. You must reason based only on the provided time series, timestamps, background information, and question.

Input Time Series: {TIME_SERIES}

Timestamps: {TIMESTAMPS}

Background Information: {BACKGROUND}

Question: {QUESTION}

Candidate Answers: True, False

Please follow the LDM reasoning procedure:

1. First describe the latent dynamic trajectory of the time series as a sequence of dynamic states, such as stable state, increasing trend, decreasing trend, fluctuation, turning point, level shift, periodic pattern, or abnormal deviation.
2. Compare the inferred latent dynamic trajectory with the statement in the question.
3. Determine whether the statement is supported or contradicted by the temporal evidence.
4. Output exactly one final answer: True or False.

Please use the following output format:

Latent Dynamic Trajectory: ...

Evidence: ...

Final Answer: <True / False>

Figure 19: Prompt template for true-or-false questions with LDM.

LDM Prompt Template for MCQs

You are an expert in time series analysis and temporal reasoning. Given a time series and a multiple-choice question, your task is to select the correct option. You must reason based only on the provided time series, timestamps, background information, question, and candidate options.

Input Time Series: {TIME_SERIES}

Timestamps: {TIMESTAMPS}

Background Information: {BACKGROUND}

Question: {QUESTION}

Candidate Options: {OPTIONS}

Please follow the LDM reasoning procedure:

1. First describe the latent dynamic trajectory of the time series as a sequence of dynamic states, such as stability, upward drift, downward drift, fluctuation, turning point, level shift, periodic pattern, or abnormal deviation.
2. Compare the latent dynamic trajectory with each candidate option.
3. Select the option that is most consistent with the inferred temporal dynamics.
4. Output exactly one final option from the candidate options.

Please use the following output format:

Latent Dynamic Trajectory: ...

Option Analysis: ...

Final Answer: <option label>

Figure 20: Prompt template for multiple-choice questions with LDM.

LDM Prompt Template for OPQs

You are an expert in time series analysis and temporal reasoning. Given a time series and an open-ended question, your task is to generate a concise answer based on the temporal evidence. You must reason based only on the provided time series, timestamps, background information, and question.

Input Time Series: {TIME_SERIES}

Timestamps: {TIMESTAMPS}

Background Information: {BACKGROUND}

Question: {QUESTION}

Please follow the LDM reasoning procedure:

1. First describe the latent dynamic trajectory of the time series as a sequence of dynamic states, such as stable state, upward drift, downward drift, fluctuation, turning point, level shift, periodic pattern, or abnormal deviation.
2. Identify the latent dynamic states that are most relevant to the question.
3. Use these dynamic states as temporal evidence to answer the question.
4. Provide a concise final answer grounded in the inferred latent dynamics.

Please use the following output format:

Latent Dynamic Trajectory: ...

Evidence: ...

Final Answer: <concise textual answer>

Figure 21: Prompt template for open-ended questions with LDM.

F.4 Implementation Details of DOM

The core idea of DOM is to formulate model outputs as a distribution over multiple candidate predictions, rather than relying on a single deterministic generation. Under a fixed sampling budget, DOM generates multiple candidate outputs and uses them to estimate output uncertainty and task-specific utility. It does not modify the underlying LLM or introduce additional training; it only changes how outputs are generated, aggregated, and selected. DOM follows a structured inference procedure based on controlled multi-sampling:

1. **Input preparation.** Given a raw time series $\mathbf{x}_{1:T}$, timestamps, optional background description, and task instruction, DOM serializes them into the same unified textual format as other reasoning strategies. No additional normalization, scaling, truncation, or resampling is applied.
2. **Controlled multi-sampling.** For each test instance, DOM generates N candidate outputs using the same prompt and decoding configuration. The sampling budget N is fixed across tasks and models, and the generated candidates are treated as samples from the model-induced output distribution.
3. **Candidate parsing.** Each sampled response is parsed by a unified parser. It extracts labels or options for discrete tasks, anomaly judgments or ranges for anomaly detection, fixed-length numerical sequences for forecasting, and valid textual answers for open-ended QA. Invalid outputs, such as malformed responses, illegal values, or length-mismatched forecasts, are discarded.
4. **Empirical distribution estimation.** For discrete-output tasks, DOM estimates the empirical probability of each candidate answer $y \in \mathcal{Y}$ as $\hat{p}(y \mid \mathbf{x}_{1:T}) = \frac{1}{N_{\text{valid}}} \sum_{i=1}^{N_{\text{valid}}} \mathbb{I}(\hat{y}^{(i)} = y)$, where N_{valid} is the number of valid sampled outputs.
5. **Uncertainty estimation.** For discrete tasks, uncertainty is computed from the empirical distribution, e.g., $\text{Unc}(\mathbf{x}_{1:T}) = 1 - \max_{y \in \mathcal{Y}} \hat{p}(y \mid \mathbf{x}_{1:T})$, or by distribution entropy. A concentrated distribution indicates low uncertainty, while a dispersed one indicates unstable decisions. For forecasting, uncertainty is measured by the sample variance

across valid predicted sequences at each future step.

6. **Distribution-aware decision.** For discrete tasks, DOM selects the answer with high empirical support and low uncertainty: $y^* = \arg \max_{y \in \mathcal{Y}} [U(y) - \lambda \text{Unc}(y)]$, where $U(y)$ denotes task-specific utility. In practice, when $U(y)$ is the empirical probability, this reduces to selecting the most stable high-probability answer. For forecasting, DOM aggregates all valid predictions by element-wise averaging: $\hat{\mathbf{x}}_{T+1:T+H} = \frac{1}{N_{\text{valid}}} \sum_{i=1}^{N_{\text{valid}}} \hat{\mathbf{x}}_{T+1:T+H}^{(i)}$.
7. **Output standardization.** The final answer is converted into the task-required format and evaluated with the same metrics as other strategies, including labels, options, anomaly intervals, fixed-length forecasts, or textual answers.

DOM Prompt Template for Classification

You are an expert in time series analysis and temporal reasoning. Given a time series, your task is to classify it into one of the candidate categories. You must reason based only on the provided time series, timestamps, background information, and task instruction.

Input Time Series: {TIME_SERIES}

Timestamps: {TIMESTAMPS}

Background Information: {BACKGROUND}

Task Instruction: {TASK_INSTRUCTION}

Candidate Labels: {LABEL_SPACE}

Please follow the DOM reasoning procedure:

1. Analyze the temporal evidence in the input time series, including trend, fluctuation, periodicity, abrupt change, stability, and abnormal deviation.
2. Consider multiple plausible candidate labels and estimate how well each candidate is supported by the observed temporal evidence.
3. Select the label that has the strongest support and the lowest uncertainty under the current evidence.
4. Output exactly one final label from the candidate labels.

Please use the following output format:

Candidate Distribution: - Candidate 1: support level / uncertainty: ... - Candidate 2: support level / uncertainty:

Decision Evidence: ...

Final Answer: <label>

Figure 22: Prompt template for classification tasks with DOM.

DOM Prompt Template for Anomaly Detection

You are an expert in time series analysis and anomaly detection. Given a time series, your task is to determine whether anomalous behavior exists and, if required, identify the anomalous time points or intervals. You must reason based only on the provided time series, timestamps, background information, and task instruction.

Input Time Series: {TIME_SERIES}

Timestamps: {TIMESTAMPS}

Background Information: {BACKGROUND}

Task Instruction: {TASK_INSTRUCTION}

Output Space: {OUTPUT_SPACE}

Please follow the DOM reasoning procedure:

1. Analyze the temporal evidence for normal and abnormal behavior, including spikes, drops, level shifts, abrupt changes, intensified fluctuation, and deviations from the expected pattern.
2. Consider multiple plausible anomaly decisions, such as normal, anomalous, point-level anomaly, or interval-level anomaly.
3. Estimate the support and uncertainty of each candidate anomaly decision based on the observed temporal evidence.
4. Select the final anomaly output that is most strongly supported and least uncertain.

Please use the following output format:

Candidate Distribution: - Candidate 1: support level / uncertainty: ... - Candidate 2: support level / uncertainty:

Anomaly Evidence: ...

Final Answer: <normal / anomalous / anomaly points / anomaly intervals>

Figure 23: Prompt template for anomaly detection tasks with DOM.

DOM Prompt Template for Forecasting

You are an expert in time series analysis and forecasting. Given a historical time series, your task is to predict its future values for the specified forecasting horizon. You must reason based only on the provided time series, timestamps, background information, and task instruction.

Input Time Series: {TIME_SERIES}

Timestamps: {TIMESTAMPS}

Background Information: {BACKGROUND}

Task Instruction: {TASK_INSTRUCTION}

Forecasting Horizon: {PREDICTION_LENGTH}

Please follow the DOM reasoning procedure:

1. Analyze the historical temporal dynamics, including trend, periodicity, fluctuation, level shift, turning point, and recent local variation.
2. Consider multiple plausible future trajectories and assess their consistency with the observed temporal evidence.
3. Estimate the uncertainty of the future dynamics based on whether the plausible trajectories are concentrated or dispersed.
4. Generate one forecast sequence that best represents the expected future values.
5. Output exactly {PREDICTION_LENGTH} numerical values.

Please use the following output format:

Candidate Future Dynamics: - Candidate trajectory 1: support level / uncertainty: ... - Candidate trajectory 2: support level / uncertainty:

Forecasting Evidence: ...

Final Answer: <v₁, v₂, ..., v_{PREDICTION_LENGTH}>

Figure 24: Prompt template for forecasting tasks with DOM.

DOM Prompt Template for TFQs

You are an expert in time series analysis and temporal reasoning. Given a time series and a true-or-false question, your task is to determine whether the statement is true or false. You must reason based only on the provided time series, timestamps, background information, and question.

Input Time Series: {TIME_SERIES}

Timestamps: {TIMESTAMPS}

Background Information: {BACKGROUND}

Question: {QUESTION}

Candidate Answers: True, False

Please follow the DOM reasoning procedure:

1. Analyze the temporal evidence that supports or contradicts the statement in the question.
2. Consider both candidate answers, True and False, and estimate their support based on the observed temporal pattern.
3. Assess the uncertainty between the two candidate answers.
4. Output exactly one final answer: True or False.

Please use the following output format:

Candidate Distribution: - True: support level / uncertainty: ... - False: support level / uncertainty: ...

Evidence: ...

Final Answer: <True / False>

Figure 25: Prompt template for true-or-false questions with DOM.

DOM Prompt Template for MCQs

You are an expert in time series analysis and temporal reasoning. Given a time series and a multiple-choice question, your task is to select the correct option. You must reason based only on the provided time series, timestamps, background information, question, and candidate options.

Input Time Series: {TIME_SERIES}

Timestamps: {TIMESTAMPS}

Background Information: {BACKGROUND}

Question: {QUESTION}

Candidate Options: {OPTIONS}

Please follow the DOM reasoning procedure:

1. Analyze the temporal evidence relevant to the question, including trend, fluctuation, periodicity, abrupt change, stability, or abnormal deviation.
2. Consider all candidate options and estimate how strongly each option is supported by the observed temporal evidence.
3. Assess the uncertainty among candidate options based on whether the evidence is concentrated on one option or dispersed across multiple options.
4. Select the option with the strongest support and lowest uncertainty.

Please use the following output format:

Candidate Distribution: - Option A: support level / uncertainty: ... - Option B: support level / uncertainty: ...

Option Analysis: ...

Final Answer: <option label>

Figure 26: Prompt template for multiple-choice questions with DOM.

DOM Prompt Template for OPQs

You are an expert in time series analysis and temporal reasoning. Given a time series and an open-ended question, your task is to generate a concise answer based on the temporal evidence. You must reason based only on the provided time series, timestamps, background information, and question.

Input Time Series: {TIME_SERIES}

Timestamps: {TIMESTAMPS}

Background Information: {BACKGROUND}

Question: {QUESTION}

Please follow the DOM reasoning procedure:

1. Analyze the temporal evidence relevant to the question, including trend, fluctuation, periodicity, abrupt change, stability, and abnormal deviation.
2. Consider multiple plausible textual answers and estimate how well each answer is supported by the observed temporal evidence.
3. Assess whether the candidate answers are consistent or uncertain under the current evidence.
4. Provide a concise final answer that is best supported by the temporal evidence.

Please use the following output format:

Candidate Answers: - Candidate answer 1: support level / uncertainty: ... - Candidate answer 2: support level / uncertainty: ...

Evidence: ...

Final Answer: <concise textual answer>

Figure 27: Prompt template for open-ended questions with DOM.

G Ablations

G.1 Disentangling Reasoning from Input Representation

To examine whether the gains of the proposed strategies come from time-aware reasoning itself or merely from better input formatting, we conduct a representation-only ablation. For each proposed strategy, we construct a corresponding representation-only variant that preserves the structured input or output format but removes the explicit reasoning procedure. **TRS-Rep** provides the segment-style organization but does not require segment-level reasoning and aggregation. **LDM-Rep** provides a latent-state-style description but does not require trajectory-level reasoning over state transitions. **DOM-Rep** keeps the same multi-sample output setting but removes the uncertainty-aware distributional decision rule. In all variants, the base LLM, input time series, decoding parameters, and output parser are kept unchanged.

The comparison is shown in Fig. 28. Across most tasks, the full time-aware reasoning strategies consistently outperform their representation-only counterparts. This indicates that the improvements cannot be fully explained by better formatting of the input sequence. Instead, the explicit reasoning protocol, including segment-level aggregation in **TRS**, trajectory-level state reasoning in **LDM**, and uncertainty-aware decision making in **DOM**, contributes substantially to the final performance. The gap is particularly clear on classification, anomaly detection, and forecasting, where task decisions strongly depend on temporal structure rather than only on surface-level input organization.

G.2 Reasoning-only Ablation

We conduct a reasoning-only ablation to separate the effect of explicit time-aware reasoning from the effect of the complete strategy design. In this setting, all reasoning-only variants receive the same raw serialized time series as **DiP**, without externally provided segment-style formatting, pre-computed latent-state summaries, or additional structured input representations. The only difference is that the prompt requires the model to follow the corresponding time-aware reasoning protocol.

TRS-R uses the raw input and asks the model to propose temporal regimes during inference, reason over local regimes, and aggregate segment-level evidence. **LDM-R** uses the raw input and asks the model to construct a latent dynamic trajectory be-

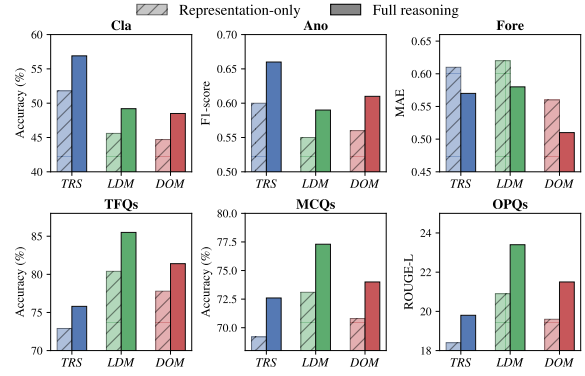


Figure 28: Representation-only ablation for disentangling the effect of input formatting from explicit time-aware reasoning.

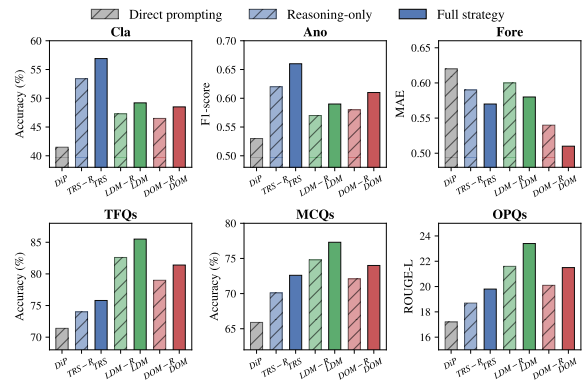


Figure 29: Reasoning-only ablation under identical raw time-series inputs. **DiP** uses direct prompting, reasoning-only variants use the same raw input but enforce time-aware reasoning protocols, and full variants correspond to the complete proposed strategies.

fore making the final decision. **DOM-R** uses the raw input and performs controlled multi-sampling with distribution-aware output selection. We compare these reasoning-only variants with both **DiP** and the full versions of **TRS**, **LDM**, and **DOM**.

As shown in Fig. 29, the reasoning-only variants generally outperform **DiP**, indicating that explicit temporal reasoning itself contributes to the gains even when the input representation is kept unchanged. Meanwhile, the full strategies further improve over the reasoning-only variants in most tasks, suggesting that the best performance is achieved when the reasoning protocol is combined with its corresponding organization and aggregation design. This confirms that the proposed methods are not merely better-formatted inputs, but structured reasoning procedures tailored to temporal decision structures.

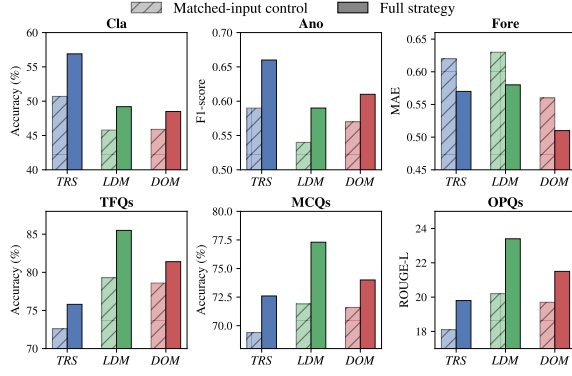


Figure 30: Matched-input control ablation. *RandSeg* uses segment-style input with non-temporal boundaries, *ShufState* uses latent-state-style input with shuffled temporal order, and *MS-only* uses the same sampling budget without uncertainty-aware decision making.

G.3 Matched-input Control Ablation

We conduct a matched-input control to examine whether the improvements simply come from more structured or compact input formatting. In this ablation, we construct control variants that preserve a similar input organization and information budget as the proposed strategies, but remove the temporal alignment of the intermediate structures.

For *TRS*, we compare it with *RandSeg*, which divides the time series into the same number of contiguous segments but uses random or uniform boundaries rather than model-inferred temporal regimes. For *LDM*, we compare it with *ShufState*, which keeps the same latent-state-style descriptions but shuffles their temporal order before reasoning. For *DOM*, we compare it with *MS-only*, which uses the same sampling budget but aggregates outputs with standard majority voting for discrete tasks and element-wise mean for forecasting, without uncertainty-aware utility selection. All variants use the same base LLM, serialized time series, decoding parameters, and output parser.

As shown in Fig. 30, the proposed strategies consistently outperform their matched-input controls. *TRS* outperforms *RandSeg*, indicating that the gain is not caused by segment formatting alone, but by aligning the segments with temporal regimes. *LDM* outperforms *ShufState*, showing that the temporal order of latent states and their transitions are essential. *DOM* outperforms *MS-only*, suggesting that controlled multi-sampling alone is insufficient and that uncertainty-aware output selection provides additional benefits. These results further support that the proposed strategies improve performance through temporally aligned reasoning,

rather than through superficial input reformatting.

H Case Study

We present a case study based on a CPU monitoring sequence to qualitatively compare different reasoning strategies under the same input and task instruction. The sequence contains multiple temporal states, including a stable period, a sharp utilization increase, and a later recovery, which makes it useful for examining whether the model can correctly preserve task-relevant temporal evidence.

Using this example, we illustrate how the three proposed time-aware strategies reason over the same sequence. *TRS* focuses on segment-level temporal regimes, *LDM* represents the sequence as a latent state trajectory, and *DOM* selects the final answer by considering the distribution and uncertainty of multiple outputs.

We further include a ToT failure case, where the selected path incorrectly shifts the decision criterion from detecting any abrupt burst to checking whether the high-utilization state persists until the end. This comparison shows that effective reasoning for time series requires alignment with the temporal decision structure, rather than simply increasing the complexity of the reasoning process.

Case Study: TRS Reasoning on a True/False Question

You are an expert in time series analysis and temporal reasoning. Given a time series and a true-or-false question, your task is to determine whether the statement is true or false. You must reason based only on the provided time series, timestamps, background information, and question.

Input Time Series:

[21.463, 22.105, 21.884, 23.017, 22.641, 24.238, 23.596, 22.917, 24.105, 25.384, 26.112, 24.763, xxx, 28.406, 29.318, 31.725, 46.892, 68.431, 72.506, 69.884, 53.271, 35.904, 30.117, 28.965]

Timestamps:

$[t_1, t_2, t_3, \dots, t_{511}]$

Background Information:

The sequence records server CPU utilization over 511 consecutive monitoring timestamps. A stable CPU utilization pattern should remain within a relatively narrow range, while a sudden increase followed by recovery indicates a burst-like workload change.

Question:

The server CPU utilization remains stable throughout the entire monitoring window, without any clear burst or abrupt increase.

Candidate Answers:

True, False

Please follow the TRS reasoning procedure:

1. Divide the complete time series into at most $\{K=3\}$ continuous temporal segments. The segments must cover the full index range from the first time step to the last time step without overlap or gaps.
2. For each segment, describe its local temporal pattern, such as stable behavior, increasing trend, decreasing trend, fluctuation, abrupt change, periodic pattern, or abnormal deviation.
3. Compare the segment-level temporal evidence with the statement in the question.
4. Output exactly one final answer: True or False.

Segments:

- Segment 1: $[t_1, t_{170}]$, local pattern: stable low utilization. The CPU values remain in a narrow range around 21–26, showing no clear burst or abrupt change.

- Segment 2: $[t_{171}, t_{390}]$, local pattern: mild fluctuation with gradual increase. The sequence still remains relatively moderate, but the utilization begins to move toward a higher range compared with the early segment.

- Segment 3: $[t_{391}, t_{511}]$, local pattern: abrupt burst followed by recovery. The visible values increase from 28.406, 29.318, and 31.725 to 46.892, 68.431, and 72.506, and then decline to 53.271, 35.904, 30.117, and 28.965.

Segment-level Evidence:

The first segment is consistent with the statement because the utilization remains stable in a low range. However, the third segment directly contradicts the statement: it contains a sharp increase to 72.506 followed by a recovery pattern. Since the question asks whether the CPU utilization remains stable throughout the entire monitoring window, the presence of one clear burst segment is sufficient to reject the statement.

Global Reasoning:

TRS aggregates the segment-level evidence by checking whether all temporal segments support the stability claim. Although the early segment appears stable, the later segment contains an abrupt workload burst. Therefore, the full sequence cannot be described as stable throughout the entire monitoring window.

Final Answer:

False

Figure 31: A case study of TRS-based reasoning on the same true-or-false question.

Case Study: LDM Reasoning on a True/False Question

You are an expert in time series analysis and temporal reasoning. Given a time series and a true-or-false question, your task is to determine whether the statement is true or false. You must reason based only on the provided time series, timestamps, background information, and question.

Input Time Series:

[21.463, 22.105, 21.884, 23.017, 22.641, 24.238, 23.596, 22.917, 24.105, 25.384, 26.112, 24.763, xxx, 28.406, 29.318, 31.725, 46.892, 68.431, 72.506, 69.884, 53.271, 35.904, 30.117, 28.965]

Timestamps:

$[t_1, t_2, t_3, \dots, t_{511}]$

Background Information:

The sequence records server CPU utilization over 511 consecutive monitoring timestamps. A stable CPU utilization pattern should remain within a relatively narrow range, while a sudden increase followed by recovery indicates a burst-like workload change.

Question:

The server CPU utilization remains stable throughout the entire monitoring window, without any clear burst or abrupt increase.

Candidate Answers:

True, False

Please follow the LDM reasoning procedure:

1. First describe the latent dynamic trajectory of the time series as a sequence of dynamic states, such as stable state, increasing trend, decreasing trend, fluctuation, turning point, level shift, periodic pattern, or abnormal deviation.
2. Compare the inferred latent dynamic trajectory with the statement in the question.
3. Determine whether the statement is supported or contradicted by the temporal evidence.
4. Output exactly one final answer: True or False.

Latent Dynamic Trajectory:

The time series can be abstracted into the following ordered latent dynamic states:

Stable low-utilization state $\rightarrow$ mild upward fluctuation $\rightarrow$ abrupt burst state $\rightarrow$ recovery state.

In the early part, the CPU utilization remains within a narrow low range, mostly around 21–26, which corresponds to a stable low-utilization state. Later, the values increase from 28.406 and 31.725 to 46.892, then further to 68.431 and 72.506, indicating an abrupt burst rather than normal fluctuation. After the burst, the sequence decreases to 53.271, 35.904, 30.117, and 28.965, forming a recovery state.

Evidence:

The statement claims that the CPU utilization remains stable throughout the entire monitoring window and does not contain any clear burst or abrupt increase. This is contradicted by the inferred latent trajectory. Although the early state is relatively stable, the later trajectory contains a clear transition from mild fluctuation to an abrupt burst, followed by recovery. Therefore, the whole sequence cannot be described as stable throughout the entire window.

Final Answer:

False

Figure 32: A case study of LDM-based reasoning on the same true-or-false question.

Case Study: DOM Reasoning on a True/False Question

You are an expert in time series analysis and temporal reasoning. Given a time series and a true-or-false question, your task is to determine whether the statement is true or false. You must reason based only on the provided time series, timestamps, background information, and question.

Input Time Series:

[21.463, 22.105, 21.884, 23.017, 22.641, 24.238, 23.596, 22.917, 24.105, 25.384, 26.112, 24.763, xxx, 28.406, 29.318, 31.725, 46.892, 68.431, 72.506, 69.884, 53.271, 35.904, 30.117, 28.965]

Timestamps:

$[t_1, t_2, t_3, \dots, t_{511}]$

Background Information:

The sequence records server CPU utilization over 511 consecutive monitoring timestamps. A stable CPU utilization pattern should remain within a relatively narrow range, while a sudden increase followed by recovery indicates a burst-like workload change.

Question:

The server CPU utilization remains stable throughout the entire monitoring window, without any clear burst or abrupt increase.

Candidate Answers:

True, False

Please follow the DOM reasoning procedure:

1. Analyze the temporal evidence that supports or contradicts the statement in the question.
2. Consider both candidate answers, True and False, and estimate their support based on the observed temporal pattern.
3. Assess the uncertainty between the two candidate answers.
4. Output exactly one final answer: True or False.

Controlled Multi-sampling:

Under the same prompt and decoding configuration, DOM generates $N = 10$ candidate outputs. After applying the unified parser, all outputs are valid:

[False, False, True, False, False, False, True, False, False, False]

Candidate Distribution:

- True: low support / high uncertainty. Two out of ten samples select True, mainly because the beginning of the sequence stays within a relatively narrow range. - False: high support / low uncertainty. Eight out of ten samples select False because the later values contain a clear abrupt increase and recovery. Equivalently, the empirical distribution is:

$$\hat{p}(\text{True} \mid \mathbf{x}_{1:T}) = 0.20, \quad \hat{p}(\text{False} \mid \mathbf{x}_{1:T}) = 0.80.$$

Uncertainty Estimation:

Using maximum-probability uncertainty, DOM obtains:

$$\text{Unc}(\mathbf{x}_{1:T}) = 1 - \max_{y \in \{\text{True}, \text{False}\}} \hat{p}(y \mid \mathbf{x}_{1:T}) = 1 - 0.80 = 0.20.$$

The low uncertainty indicates that the sampled outputs are concentrated on False.

Evidence:

The statement claims that CPU utilization remains stable throughout the entire monitoring window. However, the visible sequence contains a sharp burst in the later part: the utilization rises from 31.725 to 46.892, then further to 68.431 and 72.506, before decreasing to 53.271, 35.904, 30.117, and 28.965. This pattern contradicts the claim of stable utilization without abrupt increase.

Distribution-aware Decision:

For this discrete True/False task, DOM selects the answer with the highest empirical support and low uncertainty:

$$y^* = \arg \max_{y \in \{\text{True}, \text{False}\}} \hat{p}(y \mid \mathbf{x}_{1:T}).$$

Since $\hat{p}(\text{False} \mid \mathbf{x}_{1:T}) = 0.80$ is higher than $\hat{p}(\text{True} \mid \mathbf{x}_{1:T}) = 0.20$, the final standardized answer is False.

Final Answer:

False

Figure 33: A case study of DOM-based reasoning on a true-or-false question.

Case Study: Incorrect ToT Reasoning on a True/False Question

You are an expert in time series analysis and temporal reasoning. Given a time series and a true-or-false question, your task is to determine whether the statement is true or false. You must reason based only on the provided time series, timestamps, background information, and question.

Input Time Series:

[21.463, 22.105, 21.884, 23.017, 22.641, 24.238, 23.596, 22.917, 24.105, 25.384, 26.112, 24.763, xxx, 28.406, 29.318, 31.725, 46.892, 68.431, 72.506, 69.884, 53.271, 35.904, 30.117, 28.965]

Timestamps:

$[t_1, t_2, t_3, \dots, t_{511}]$

Background Information:

The sequence records server CPU utilization over 511 consecutive monitoring timestamps. A stable CPU utilization pattern should remain within a relatively narrow range, while a sudden increase followed by recovery indicates a burst-like workload change.

Question:

The server CPU utilization remains stable throughout the entire monitoring window, without any clear burst or abrupt increase.

Candidate Answers:

True, False

Please follow the ToT reasoning procedure:

1. Generate multiple candidate reasoning paths for the given time series and question.
2. Evaluate each path according to its consistency with the observed temporal evidence.
3. Select the most plausible path.
4. Output exactly one final answer: True or False.

Candidate Reasoning Paths:

Path 1: Overall-stability path. Most visible values are within a low-to-moderate utilization range. The beginning of the sequence remains around 21–26, and the final values decrease back to around 30. Since the sequence does not remain at a high level until the end, the temporary high values can be interpreted as short-term fluctuation rather than a persistent instability.

Path 2: Burst-detection path. The later part of the sequence contains a sharp increase from 31.725 to 46.892, 68.431, and 72.506. This rise is much larger than the earlier low-range fluctuation. The subsequent decrease to 53.271, 35.904, 30.117, and 28.965 indicates recovery after a burst. This path contradicts the statement that there is no abrupt increase.

Path 3: Recovery-weighted path. The sequence contains a temporary elevation, but the final observations return close to the previous range. Since the high values are not sustained and the signal recovers, the overall monitoring window may still be regarded as stable with a transient perturbation.

Path Evaluation:

Path 1 and Path 3 both emphasize that the sequence begins and ends in a relatively moderate range. Path 2 identifies a sharp local increase, but this increase is followed by recovery. Since the question concerns the whole monitoring window, the recovery pattern is treated as evidence that the sequence does not undergo persistent instability. Therefore, the recovery-weighted interpretation is selected.

Selected Path:

Path 3: Recovery-weighted path.

Final Answer:

True

Failure Analysis:

The selected ToT path is incorrect because one branch shifts the decision criterion from “whether there exists any clear burst or abrupt increase” to “whether the high-utilization state persists until the end.” During the tree search, this branch appears plausible because the sequence eventually recovers to a lower level. However, it overlooks the original condition that the sequence should remain stable throughout the entire monitoring window. The sharp rise from 31.725 to 46.892, 68.431, and 72.506 already constitutes an abrupt increase, regardless of the subsequent recovery. Therefore, this ToT path selects an incorrect branch, and the correct answer should be False.

Figure 34: A failure case of ToT reasoning on the same true-or-false question.