SenTSR-Bench: Thinking with Injected Knowledge for Time-Series Reasoning

Zelin He[1†] **Boran Han**[2] **Xiyuan Zhang**[2] **Shuai Zhang**[2] **Haotian Lin**[3]

Qi Zhu[2] **Haoyang Fang**[2] **Danielle C. Maddix**[2] **Abdul Fatir Ansari**[2]

Akash Chandrayan[3] **Abhinav Pradhan**[3] **Bernie Wang**[2] **Matthew Reimherr**[1,3]

[1]The Pennsylvania State University [2]AWS AI Labs [3]Amazon RME

Abstract

Time-series diagnostic reasoning is essential for many applications, yet existing solutions face a persistent gap: general reasoning large language models (GRLMs) possess strong reasoning skills but lack the domain-specific knowledge to understand complex time-series patterns. Conversely, fine-tuned time-series LLMs (TSLMs) understand these patterns but lack the capacity to generalize reasoning for more complicated questions. To bridge this gap, we propose a hybrid *knowledge-injection* framework that injects TSLM-generated insights directly into GRLM's reasoning trace, thereby achieving strong time-series reasoning with in-domain knowledge. As collecting data for knowledge injection fine-tuning is costly, we further leverage a reinforcement learning-based approach with verifiable rewards (RLVR) to elicit knowledge-rich traces *without human supervision*, then transfer such an in-domain thinking trace into GRLM for efficient knowledge injection. We further release *SenTSR-Bench*, a multivariate time-series-based diagnostic reasoning benchmark collected from *real-world industrial operations*. Across *SenTSR-Bench* and other public datasets, our method consistently surpasses TSLMs by 9.1%–26.1% and GRLMs by 7.9%–22.4%, delivering robust, context-aware time-series diagnostic insights.

Proceedings of the 29[th] International Conference on Artificial Intelligence and Statistics (AISTATS) 2026, Tangier, Morocco. PMLR: Volume 300.

1 INTRODUCTION

Diagnostic reasoning over time-series data is a fundamental capability in many domains, enabling critical tasks such as event characterization, root-cause diagnosis, and decision-making (Leite et al., 2024; Chen et al., 2024a). In industrial operations, for instance, streams of sensor data measuring machine temperature and vibration are analyzed to diagnose potential equipment failures (Figure 1 (a)). However, existing research in this domain has predominantly focused on surface-level anomaly detection (Alnegheimish et al., 2025). While effective at identifying irregularities, these techniques cannot offer actionable insights because they lack the capacity for temporal and causal reasoning required to explain an anomaly's origin, diagnose its root cause, or recommend corrective actions.

Recent advances in LLMs (Jaech et al., 2024; Guo et al., 2025; Anthropic, 2025), have unlocked an enhanced reasoning capabilities via embed implicit reasoning mechanisms (Yeo et al., 2025), yielding remarkable gains on benchmarks requiring reasoning. However, these general reasoning LLMs (GRLMs) lack the domain knowledge needed to interpret complex time-series patterns, thereby producing incorrect reasoning trajectories and thus incorrect diagnoses (Merrill et al., 2024; Cao et al., 2026). In parallel, smaller LLM variants fine-tuned on domain-specific time-series-textual pairs (Xie et al., 2024; Zhang et al., 2025) have shown improved alignment with time-series understanding tasks. Yet these fine-tuned time-series language models (TSLMs) frequently overfit to narrow template, like tasks and lack the reasoning depth or generalization capacity required for out-of-distribution scenarios. As a result, *both standalone GRLMs and TSLMs fall short in practice* (illustrated in Figure 1(c)).

[†]Work done during an internship at Amazon.

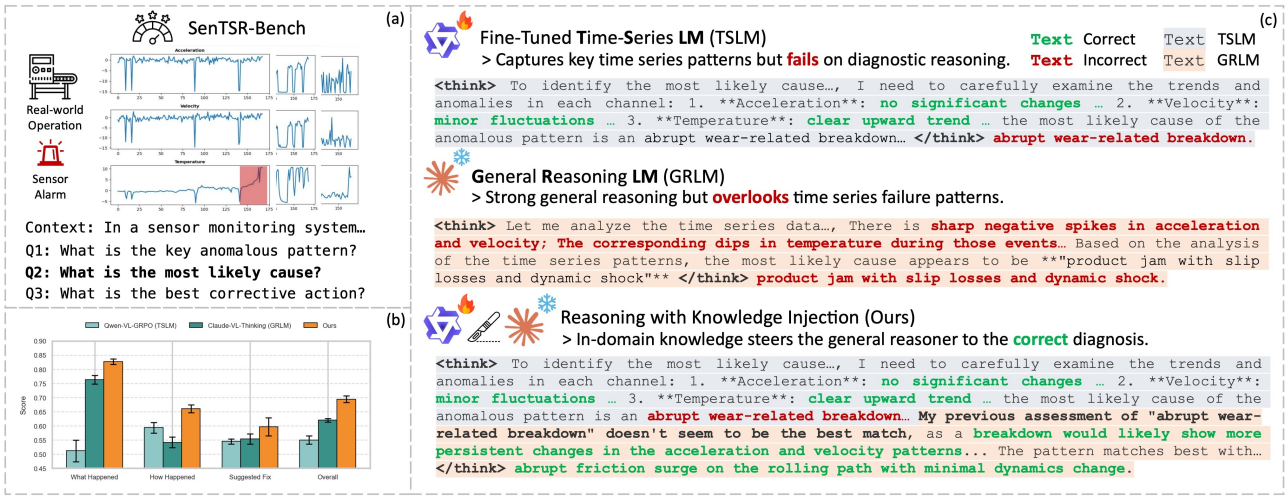


Figure 1: (a) The newly released *SenTSR-Bench* benchmark, collected from real-world machine monitoring environments, with multi-stage diagnostic questions. (b) Performance of the proposed framework on *SenTSR-Bench*, surpassing both stand-alone time-series specialists (TSLM) and general reasoning models (GRLM). (c) Case study illustrating why knowledge injection helps: the *specialist* captures key time-series patterns but fails to connect them to the correct root cause; the *general reasoner* shows strong reasoning but overlooks domain-specific critical failure patterns; our method injects the *in-domain knowledge* from fine-tuned specialist into the reasoner's *reasoning trace*, aligning the trace with domain knowledge and producing the correct diagnosis.

To address the above challenge, we propose a *reasoning with knowledge injection* framework that couples the reasoning power of GRLMs with the in-domain knowledge of TSLMs. At its core, the framework injects knowledge from TSLMs directly into the reasoning process of GRLMs, allowing the generated reasoning trace to continue with guidance from in-domain information. When the injected knowledge is reliable, it helps steer the reasoning trajectory toward accurate diagnoses; when the knowledge is weaker, the model corrects it with its strong critical thinking capacity.

One additional challenge is that a TSLM trained for in-domain question answering often fails to function effectively as an assistant for a GRLM. A typical alternative is to finetune a dedicated helper model, but this approach is constrained by the need to construct large, high-quality datasets explicitly tailored for knowledge injection. To overcome this supervision bottleneck, we introduce thinking transfer. Our method trains the TSLM within a reinforcement learning with verifiable reward framework (Guo et al., 2025), leveraging rule-based verifiable rewards and an explicit thinking structure to naturally elicit knowledge-rich thinking traces without any manual supervision. At inference, these RL-honed traces are injected into the GRLM, providing it with high-quality, in-domain knowledge to ground its subsequent reasoning process.

Furthermore, to benchmark time-series diagnostic reasoning in real-world diagnostic settings, we introduce Sensor-based Time-Series Diagnostic Reasoning (*SenTSR-Bench*) Benchmark, a first-of-its-kind dataset of multivariate sensor streams and diagnostic texts for time-series diagnostic reasoning evaluation. In contrast to prior benchmarks that are either purely synthetic or LLM-annotated, *SenTSR-Bench* is built on the *real-world multivariate time-series data* drawn from real-world diagnostics events with *human-annotated data*.

Across *SenTSR-Bench* and other existing benchmark datasets, and on both closed-source and open-source reasoning models, our method surpasses TSLMs by 9.1–26.1% and GRLMs by 7.9–22.4%. RL-enhanced injection further yields 1.66×–2.92× larger gains than SFT-enhanced injection, and consistently outperforms few-shot prompting and prompt-based collaboration approaches. Taken together, our key contributions are as follows:

- **New Paradigm for Time-series Reasoning.** We formalize a framework that injects in-domain knowledge from a TSLM into an GRLM's reasoning process, steering reasoning with domain knowledge.
- **RL-Based Method for Efficient Injection.** We propose an injection paradigm that utilizes reinforcement learning with verifiable rewards to elicit knowledge-rich thinking traces *without manual supervision* for injection.
- **Real-World Benchmark and Evaluation.** We release *SenTSR-Bench*, a de-identified, real-world mul-

tivariate time-series benchmark for diagnostic reasoning. Evaluations on *SenTSR-Bench* and public datasets show state-of-the-art diagnostic accuracy of our proposed solution with interpretable explanations.

2 METHODOLOGY

Figure 2 provides an overview of our proposed framework. In this section, we first establish preliminaries and formally define the reasoning model generation process (Section 2.1). We then introduce the general paradigm of knowledge injection (Section 2.2). We then further instantiate this framework (Section 2.3). Finally, we describe a reinforcement learning-based framework for efficient knowledge injection (Section 2.4).

2.1 Preliminaries and Notation

Multimodal Input. Write V for the discrete token vocabulary and V^* for the space of finite token sequences, and use $[a, b]$ to denote the concatenation of two sequences a and b. Let $\mathbf{q} = (q_1, \ldots, q_n) \in V^*$ be a sequence of textual tokens describing the task (e.g., question, context, or instructions). A multivariate time-series is denoted by $\mathbf{X} = \{\mathbf{x}_t\}_{t=1}^{T}$, where each $\mathbf{x}_t \in \mathbb{R}^D$ is the reading of D channels at time step t. To interface with language models, $\mathbf{X}$ must be mapped into the token space V^*. This can be done, for example, by rendering the series as a line-plot image and encoding it (Liu et al., 2025c), converting it into structured JSON text followed by standard text tokenization, or applying a specialized time-series tokenizer (Xie et al., 2024). With slight abuse of notation, we use $\mathbf{X}$ to denote the final tokenized representation.

Reasoning Model. We define a reasoning model through its generative distribution π (also referred to as a policy in later context) that generates two outputs: an internal reasoning trace $\mathbf{r} = (r_1, \ldots, r_K) \in V^*$ and a final answer $\mathbf{y} = (y_1, \ldots, y_M) \in V^*$. Generation proceeds in two phases. In the reasoning phase, the model autoregressively produces a latent reasoning trace conditioned on the input pair $(\mathbf{X}, \mathbf{q})$ and a special thinking structure:

$$\pi(\mathbf{r} \mid \mathbf{X}, \mathbf{q}) = \prod_{k=1}^{K} \pi(r_k \mid \mathbf{X}, \mathbf{q}, [\langle\text{think}\rangle, \mathbf{r}_{<k}]). \quad (1)$$

Here, $\langle$think$\rangle$ marks the beginning of the reasoning segment, which continues until the model emits the closing token $\langle$/think$\rangle$. In the response phase, the model conditions on both the input and the full reasoning trace to generate the final answer:

$$\pi(\mathbf{y} \mid \mathbf{X}, \mathbf{q}, [\langle\text{think}\rangle, \mathbf{r}, \langle/\text{think}\rangle]) = \prod_{j=1}^{M} \pi\Big(y_j \,\Big|\, \mathbf{X}, \mathbf{q}, [\langle\text{think}\rangle, \mathbf{r}, \langle/\text{think}\rangle, \mathbf{y}_{<j}]\Big). \quad (2)$$

This reasoning–then–response decomposition exposes the latent reasoning trace $\mathbf{r}$, which we later *inspect* and *modify* through knowledge injection.

In this paper, we distinguish two models. A (frozen) general reasoning model (GRLM), quantified by π^G, is a large open/closed–source model that follows the reasoning–then–response factorization discussed above, and a time-series language model (TSLM), quantified by π^T, is a small fine–tuned in-domain specialist.

2.2 General Knowledge Injection Paradigm

Specialist Knowledge Generation. Given the current reasoning state of the general reasoner π^G at step k, i.e., the prefix $\mathbf{r}^G_{<k}$ together with inputs $(\mathbf{X}, \mathbf{q})$, we form an *injection–oriented token sequence* $\tilde{\mathbf{q}} = \mathsf{Query}(\mathbf{q}, \mathbf{r}^G_{<k})$, where $\mathsf{Query}(\cdot)$ is a deterministic query–shaping function (e.g., "provide helpful information", "validate the claims"). Concrete choices are given in later subsections. Then a TSLM is invoked on $(\mathbf{X}, \tilde{\mathbf{q}})$ to produce an output sequence

$$\mathbf{K}^T \sim \pi^T(\cdot \mid \mathbf{X}, \tilde{\mathbf{q}}).$$

Intuitively, $\mathbf{K}^T$ stands for the relevant in-domain time-series knowledge for injection.

Reasoning with Knowledge Injection. Given the TSLM knowledge output $\mathbf{K}^T$ and the current GRLM reasoning prefix $\mathbf{r}^G_{\le k}$, we apply injection with

$$\mathbf{r}^{\text{Inj}}_{\le k} = \mathsf{Inject}(\mathbf{r}^G_{\le k}, \mathbf{K}^T),$$

which returns an updated thinking trace prefix to be used. Here $\mathsf{Inject}(\cdot)$ is a deterministic injection function; the choice of k is chosen based on the injection method. The general reasoner GRLM then resumes generation conditioned on the updated prefix:

$$r^G_j \sim \pi^G\Big(\cdot \,\Big|\, \mathbf{X}, \mathbf{q}, [\langle\text{think}\rangle, \mathbf{r}^{\text{Inj}}_{<j}]\Big), \qquad j \ge k, \quad (3)$$

and then produce the final answer similar to Eq. (2).

2.3 Instantiating Knowledge Injection

Early Knowledge Injection A simple yet effective way to realize the injection paradigm is through early injection: immediately after $\langle$think$\rangle$. We choose a tokenized instruction $\mathbf{v}_{\text{help}}$ (e.g., "produce a step–by–step

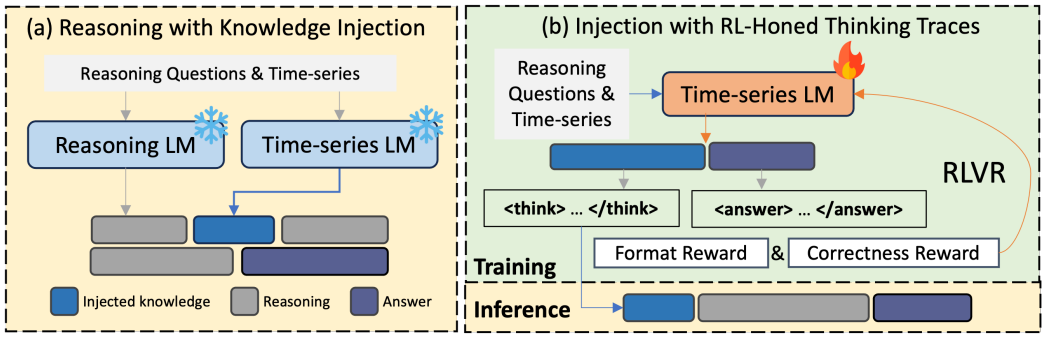


Figure 2: Overview of the proposed paradigm. (a) *Knowledge injection:* given a reasoning question and its time-series, a time–series LM (TSLM) produces grounded analysis snippets that are injected into the reasoning trace of a general *frozen* reasoning LM (GRLM) to answer diagnostic queries without weight updates. (b) *Thinking transfer via RL:* We train the TSLM using reinforcement learning with *verifiable rewards* (RLVR) with an explicit thinking structure to *elicit* analysis-first thinking traces *without human supervision*; at inference, these traces are transferred via injection into the reasoning LM to strengthen temporal grounding for diagnosis.

analysis of the question with the time-series data") and form

$$\tilde{\mathbf{q}} \;=\; \mathsf{Query}_{\text{help}}(\mathbf{q},\,\emptyset) = [\mathbf{q}, \mathbf{v}_{\text{help}}]\,.$$

The specialist then generates the knowledge snippet $\mathbf{K}^T \sim \pi^T(\cdot \mid \mathbf{X}, \tilde{\mathbf{q}})$ from the learnt time-series knowledge, which we then appended a brief reflection trigger to elicit critical reasoning $\mathbf{v}_{reflect}$ (e.g., "Wait, let me reflect on my previous thinking process with the time-series data."). We then inject at k=1:

$$\mathbf{r}^{\text{Inj}}_{\leq 1} \;=\; \mathsf{Inject}_{reflect}\big(\emptyset,\; \mathbf{K}^T\big) = \big[\,\mathbf{K}^T, \boldsymbol{v}_{reflect}\,\big],$$

after which the general reasoner π^G continues its reasoning trace and produces the final response conditioned on $\mathbf{r}^{\text{Inj}}_{\leq 1}$ (cf. Section 2.1). Conceptually, π^T contributes *grounded, in–domain time-series-based insights* extracted from $\mathbf{X}$, while π^G performs the *general reasoning* by integrating the injected knowledge with context $\mathbf{q}$, adjudicating alternatives, and producing the final answer.

Other Injection Paradigms Beyond early injection, the framework also supports alternative strategies. Examples include *intermediate injection* that corrects the GRLM's reasoning process by inserting TSLM's knowledge at low-confidence points in the reasoning trace; or *late injection* that prompts TSLM to critique the entire GRLM reasoning trace and prompts reflection before the final answer. Full implementation details are provided in Appendix E. In practice, we find that early injection is the most broadly effective, and thereby we adopt early injection as the default in subsequent method development, and report comparison results for the other variants in Section 4.3.

Practical Implementation. The method is easy to implement and compatible with standard LLM APIs.

Algorithm 1: Algorithm Workflow for Knowledge Injection with RL Honed Thinking

Input: Training set $\mathcal{D}_{\text{train}}$, test set $\mathcal{D}_{\text{test}}$, general reasoner policy π^G

Output: Trained specialist policy π^T and predictions on $\mathcal{D}_{\text{test}}$

```
# Stage I: Train TSLM with RLVR
for (X, q, y*) ∈ D_train do
    Update π^T using RLVR training with composite reward   // cf. Eq. (6)
# Stage II: Inference-time knowledge injection for GRLM
for (X, q) ∈ D_test do
    Obtain r^T ~ π^T(· | X, q)
    Form r^Inj_≤1 ← Inject_reflect(∅, r^T)   // cf. Eq. (5)
    Obtain r^G ~ π^G(· | X, q, [⟨think⟩, r^Inj_≤1])
    Produce and record y^G with X, q, r^G   // cf. Eq. (2)
return π^T and all test predictions
```

For models that support assistant prefill, the injected trace can be directly fed as assistant's initial tokens by pre-inserting $[\langle\text{think}\rangle,\ \mathbf{r}^{\text{Inj}}_{\leq k}]$. For models do not allow prefill for reasoning traces, we instead use an instructional proxy by wrapping the injected trace in the model's recommended thinking templates. See Appendix E for details.

2.4 Knowledge Injection with RL-Honed Thinking Traces

A time-series specialist π^T is typically optimized for direct question answering,

$$\mathbf{y}^T \sim \pi^T(\cdot \mid \mathbf{X}, \mathbf{q}),$$

where the objective is to predict the answer tokens $\mathbf{y}^T$ given inputs $(\mathbf{X}, \mathbf{q})$. In contrast, knowledge injection requires the specialist to provide an intermediate analysis or evidence rather than a final answer. This is usually elicited through a help-oriented query,

$$\mathbf{K}^T \sim \pi^T(\cdot \mid \mathbf{X}, \tilde{\mathbf{q}}), \quad \tilde{\mathbf{q}} = [\mathbf{q}, \mathbf{v}_{\text{help}}],$$

where recall $\mathbf{v}_{\text{help}}$ is an instruction for producing helping knowledge (cf. Section 2.3). This mismatch induces a *task shift*: the TSLM π^T, trained to produce direct answers, tends to generate hallucinated content rather than faithful, unbiased analysis. As a result, $\mathbf{K}^T$ is systematically misaligned with the desired ground-truth knowledge for injection. Constructing large expert-annotated corpora specifically for this injection setting could mitigate the issue but is prohibitively costly.

Thinking Transfer. To resolve the task shift between answering and supplying knowledge, we propose to align the specialist with its injection role by training it to produce a thinking trace before any answer. Then, such a specialist thinking trace is served directly as the knowledge source,

$$\mathbf{K}^T_{think} := \mathbf{r}^T \sim \pi^T(\cdot \mid \mathbf{X}, \mathbf{q}). \tag{4}$$

At inference, we perform injection by starting GRLM reasoning with this analysis and a brief reflection cue,

$$\mathbf{r}^{\text{Inj}}_{\leq 1} = \mathsf{Inject}_{reflect}(\emptyset, \mathbf{r}^T) = [\mathbf{r}^T, \boldsymbol{v}_{\text{reflect}}], \tag{5}$$

and then continue the general reasoning process. This design naturally aligns training and deployment: the TSLM learns to produce analysis first, and the injected analysis serves as a grounded knowledge source for steering the reasoner.

RL Training without Thinking Supervision. Directly training a TSLM to produce analysis-first thinking traces, as in Eq. (4), is challenging as most time-series diagnostic datasets contain only ground-truth answers $\mathbf{y}^*$ but not the intermediate reasoning traces $\mathbf{r}^*$. To overcome this, we employ reinforcement learning with verifiable rewards (RLVR) (Guo et al., 2025). Let $\mathbf{z} = [\mathbf{r}, \mathbf{y}]$ denote a sampled completion containing both a trace and an answer. For each context

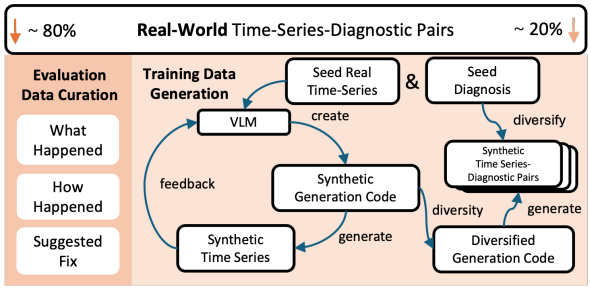


Figure 3: SenTSR-Bench Construction pipeline.

$(\mathbf{X}, \mathbf{q})$, we draw a group of G completions $\{\mathbf{z}_i\}_{i=1}^G$ and optimize with the group-relative objective:

$$\max_\theta \mathbb{E}_{\{\mathbf{z}_i\}_{i=1}^G \sim \pi_\theta(\cdot|\mathbf{X},\mathbf{q})} \left[\mathcal{L}_{GRPO}\big(\theta, \{R(\mathbf{z}_i)\}_{i=1}^G\big) \right], \tag{6}$$

with $R(\mathbf{z}) = r_{\text{fmt}}(\mathbf{z}) + r_{\text{hard}}(\mathbf{z})$, where $r_{\text{fmt}} \in \{0, 1\}$ is a format reward that equals 1 if the output follows the target structure

$$\langle \text{think} \rangle \, \mathbf{r} \, \langle /\text{think} \rangle \, \langle \text{answer} \rangle \, \mathbf{y} \, \langle /\text{answer} \rangle,$$

and 0 otherwise. The hard reward $r_{\text{hard}} \in \{0, 1\}$ equals 1 if the predicted answer $\mathbf{y}$ matches the ground-truth $\mathbf{y}^*$, and 0 otherwise. Here the objective is computed over groups of G sampled completions, with rewards normalized within the group; the detailed form of $\mathcal{L}_{GRPO}$ is provided in Appendix A.1. Importantly, *no labeled traces are required*: the policy is driven to *elicit* analysis-first reasoning purely through structural and correctness feedback. This is particularly valuable for time-series diagnostics, where ground-truth outcomes are available but the intermediate causal links between the time seires $\mathbf{X}$ and the underlying root cause $\mathbf{y}^*$ is unobserved and must be discovered through learning. The full algorithm is summarized in Algorithm 1.

3 BENCHMARK: SENTSR-BENCH

Table 1: Comparison of time-series diagnostic reasoning benchmarks.

Benchmark	New Time-Series?	Real-World?	Multi-stage Advancing Questions?	Annotation?
TSEvol (Xie et al., 2024)	✗	✓	✗	LLM
TS&Language (Merrill et al., 2024)	✓	✗	✗	LLM
MTBench (Chen et al., 2025b)	✓	✓	✗	LLM
Sensor-TSR (Ours)	✓	✓	✓	Human

Despite the growing interest in time-series diagnostic reasoning, there are still very limited high-quality datasets that couple real-world time-series with textual diagnostic annotations. As summarized in Table 1, existing work primarily rely on LLM-annotated versions of public time-series datasets or fully synthetic

time series–text pairs, and typically provide only a single question per series, falling short of capturing real-world diagnostic complexity. In this work, we introduce *SenTSR-Bench*, a new benchmark directly motivated by real-world sensor monitoring for machine breakdown diagnosis and troubleshooting. The benchmark consists of de-identified, multivariate time-series signals collected from vibration (acceleration, velocity) and temperature sensors, paired with human-curated diagnostic annotations.

SenTSR-Bench moves beyond anomaly flagging and evaluate the full procedure of diagnostic reasoning. The benchmark contains different levels of questions: (i) what happened (recognizing anomalous segments in multivariate time-series), (ii) how happened (inferring plausible root causes behind the observed signals), and (iii) suggested fix (proposing potential corrective actions). This benchmark provides a realistic and challenging testbed for developing models capable of robust, context-aware diagnostic reasoning. Figure 3 shows a simplified version of the data construction pipeline. Additional details on the benchmark construction pipeline is provided in Appendix D.

Evaluation Dataset Curation To build the evaluation dataset, we follow a three-stage curation pipeline. First, we filter 110 multivariate sensor streams out of an initial pool of over 2,000 candidate samples, selecting those that exhibit clear anomalous patterns tied to potential troubleshooting actions. All signals are then standardized to remove sensitive information. Second, we design an annotation pipeline that generates multi-stage diagnostic text while preserving privacy, producing faithful but de-identified annotations. Third, we construct 330 multiple-choice questions (MCQ) by pairing ground-truth answers with distractors. This process yields a benchmark that is both realistic and privacy-preserving, while supporting rigorous evaluation of anomaly recognition, root cause reasoning, and fix proposal tasks.

Training Dataset Generation. A key challenge is generating diverse *multivariate* sensor streams with a small number of real-world seeds are available. To address this, we design a two-stage synthetic generation pipeline powered by vision–language models (VLMs). *Stage 1: Iterative code synthesis* prompts a VLM with plots and context from 23 de-identified seeds to produce Python codes that mimic the original behaviors. *Stage 2: Diversification and simplification* transforms these simulators into compact stochastic generators that introduce randomized dynamics and parameter variation, yielding broad families of realistic synthetic series. The resulting synthetic data are then used to construct 6,000 MCQ training entries consistent with the evaluation design.

4 EXPERIMENTS

4.1 Experiment Setup

Datasets. For evaluation, we use *SenTSR-Bench*, our de-identified, real-world benchmark of multivariate time-series with three progressively harder tasks: *What happened* (key time-series anomaly characterization), *How it happened* (root-cause diagnosis), and *Suggested fix* (action recommendation). We additionally assess the performance on two public benchmarks: *TSEvol* (Dataset A) from Xie et al. (2024), which covers *inductive*, *deductive*, and *causal* reasoning, and *MCQ2* dataset from *TS&Language* Benchmark (Merrill et al., 2024), which poses relational queries over paired time-series under textual context. Additional details on the datasets are provided in Appendix D.

Implementation and Evaluation For the general reasoning model, we test the open-source models `DeepSeekR1-Distilled-Qwen-32B` (Guo et al., 2025) and `Qwen3-32B` (Yang et al., 2025) as well as closed-source models `Claude3.7` (Anthropic, 2025) with time-series encoded as either the vision form (`-vision`) or the textual form (`-text`). All models are set up with standard config. For fine-tuned TSLM, we primarily use `Qwen2.5-VL-3B` (Bai et al., 2025) for SFT and RL training. We also use `ChatTS-14B` (Xie et al., 2024) for injection design exploration. For evaluation, generative QA tasks (inductive reasoning in *SenTSR-Bench*) are evaluated using RAGAS. Verifiable tasks report *accuracy*. All results are averaged over three independent runs. Further details on implementation and evaluation are provided in Appendix E.

4.2 Performance Analysis

For performance analysis, we evaluate the proposed knowledge injection framework on both our newly released *SenTSR-Bench* benchmark and public benchmarks. We test injection across different TSLM training and injection paradigms (SFT/RL-based), and multiple general reasoning models. Results are presented in Table 1. Here are the observations:

Injection Lifts Both Baselines. Across all benchmark datasets, injecting TSLM knowledge, whether from SFT or RL-tuned TSLM, consistently boosts accuracy over both stand-alone specialists and stand-alone reasoners. On *SenTSR-Bench*, gains range from +15.5% to +26.1% over the specialized TSLM and +7.3% to +22.4% over the general GRLM; improvements span all three tasks and are most pronounced

Table 2: Reasoning performance on *SenTSR-Bench* Benchmark (mean±std). Best per block are **bolded**. The last two columns report relative gains (in %) for Injection rows vs. the corresponding specialized TSLM and the zero-shot general reasoner (GRLM), respectively.

Model	Paradigm	What Happened	How Happened	Suggested Fix	**Overall**	**Improvement vs.**	
						TSLM	**GRLM**
TSLM (Qwen-VL-3B)	SFT	0.530 ± 0.037	0.567 ± 0.029	0.548 ± 0.011	0.549 ± 0.019	—	—
	RL	0.512 ± 0.038	0.594 ± 0.019	0.546 ± 0.009	0.551 ± 0.014	—	—
GRLM (Claude3.7-Text)	Zero-shot	0.712 ± 0.019	0.409 ± 0.033	0.473 ± 0.024	0.531 ± 0.011	—	—
	Few-shot	0.691 ± 0.009	0.561 ± 0.011	0.509 ± 0.009	0.587 ± 0.006	—	+10.5%
TSLM + GRLM	SFT-Injection	0.742 ± 0.023	0.603 ± 0.021	**0.558 ± 0.019**	0.634 ± 0.006	+15.5%	+19.4%
	RL-Injection	**0.779 ± 0.014**	**0.627 ± 0.018**	0.542 ± 0.028	**0.650 ± 0.010**	+18.0%	+22.4%
TSLM (Qwen-VL-3B)	SFT	0.530 ± 0.037	0.567 ± 0.029	0.548 ± 0.011	0.549 ± 0.019	—	—
	RL	0.512 ± 0.038	0.594 ± 0.019	0.546 ± 0.009	0.551 ± 0.014	—	—
GRLM (Claude3.7-Vision)	Zero-shot	0.764 ± 0.016	0.542 ± 0.019	0.555 ± 0.018	0.620 ± 0.006	—	—
	Few-shot	0.824 ± 0.014	0.552 ± 0.014	0.555 ± 0.018	0.643 ± 0.005	—	+3.7%
TSLM + GRLM	SFT-Injection	0.756 ± 0.031	0.588 ± 0.013	**0.649 ± 0.029**	0.665 ± 0.020	+21.1%	+7.3%
	RL-Injection	**0.827 ± 0.009**	**0.661 ± 0.014**	0.597 ± 0.032	**0.695 ± 0.012**	+26.1%	+12.1%

Table 3: Reasoning performance on *TSEvol* and *TS&Language* Benchmark (mean±std). Best per block are **bolded**. The last two columns report relative gains (in %) for Injection rows vs. the corresponding specialized TSLM and the zero-shot general reasoner (GRLM), respectively.

Model	Paradigm	Causal	Deductive	Inductive	MCQ2	**Overall**	**Improvement vs.**	
							TSLM	**GRLM**
TSLM (Qwen-VL-3B)	SFT	0.623 ± 0.006	0.520 ± 0.013	0.357 ± 0.010	0.507 ± 0.032	0.502 ± 0.005	—	—
	RL	**0.627 ± 0.016**	0.496 ± 0.014	0.313 ± 0.023	**0.597 ± 0.031**	0.508 ± 0.006	—	—
GRLM (Qwen3-32B)	Zero-shot	0.507 ± 0.041	0.473 ± 0.035	**0.623 ± 0.036**	0.407 ± 0.015	0.502 ± 0.023	—	—
	Few-shot	0.622 ± 0.028	0.473 ± 0.035	0.460 ± 0.033	0.427 ± 0.015	0.495 ± 0.010	—	-1.4%
TSLM+GRLM	SFT-Injection	0.569 ± 0.035	**0.543 ± 0.013**	0.592 ± 0.031	0.410 ± 0.036	0.528 ± 0.008	+5.2%	+5.2%
	RL-Injection	**0.627 ± 0.025**	0.512 ± 0.047	0.588 ± 0.035	0.490 ± 0.046	**0.554 ± 0.021**	+9.1%	+10.4%
TSLM (Qwen-VL-3B)	SFT	0.623 ± 0.006	0.520 ± 0.013	0.357 ± 0.010	0.507 ± 0.032	0.502 ± 0.005	—	—
	RL	0.627 ± 0.016	0.496 ± 0.014	0.313 ± 0.023	**0.597 ± 0.031**	0.508 ± 0.006	—	—
GRLM (R1-Distilled-Qwen-32B)	Zero-shot	0.522 ± 0.022	0.550 ± 0.054	0.525 ± 0.015	0.483 ± 0.015	0.520 ± 0.010	—	—
	Few-shot	0.542 ± 0.017	**0.558 ± 0.040**	0.478 ± 0.022	0.513 ± 0.021	0.523 ± 0.007	—	+0.6%
TSLM+GRLM	SFT-Injection	0.594 ± 0.023	0.535 ± 0.023	0.519 ± 0.004	0.490 ± 0.020	0.534 ± 0.007	+6.4%	+2.7%
	RL-Injection	**0.634 ± 0.013**	0.543 ± 0.013	**0.532 ± 0.010**	0.537 ± 0.032	**0.561 ± 0.011**	+10.4%	+7.9%

on *How happened*, which involves both in-domain anomaly detection knowledge and strong causal reasoning capacity. On the public benchmarks, we observe similar trends: +5.2% to +10.4% over the specialist and +2.7% to +10.4% over the reasoner. The injected variant shows robustness: even when the TSLM performs poorly (e.g., the *Inductive* task), the injected model leverages the reasoner's critical thinking capacity to maintain competitive performance. Taken together, injection delivers the best *overall* performance across settings.

RL-based Injection Consistently Yields Larger Gains. Compared with SFT-based injection, RL-based *thinking transfer* delivers consistently larger improvements over zero-shot GRLMs: when measuring gains, RL-based injection provides 1.66× the improvement on `Claude3.7-Vision`, 2.00× on `Qwen3-32B`, and 2.92× on `DeepSeekR1-Distilled-Qwen-32B`. Both SFT and RL injection outperform few-shot prompting, but RL provides the biggest lifts (e.g., 3.27× than few-shot on `Claude3.7-Vision`). Moreover, injection is more *token-efficient*: while tokenized multivariate time-series in *TSevol* can exceed ∼ 50k tokens, making few-shot prompts infeasible, injection instead provides a compact analysis snippet through thinking prefill, offering a more scalable mechanism for time-series diagnostic reasoning.

4.3 Framework Analysis

Comparison across Different Injection Strategies. We evaluate three *injection strategies*—*early*, *intermediate*, and *late*. *Early injection* inserts the specialist's analysis immediately after the opening token, *Intermediate injection* with correcting the lowest-confidence token position and *Late injection* appends a specialist-generated critique to the full reasoning trace at the end. Further implementation details are provided in Appendix E. We use `ChatTS-14B` (Xie et al., 2024) here (rather than smaller specialists) as it is tuned with a broad range of time-series QA tasks and thus better trained to investigate the problem. Table 3 reports results across `Claude3.7-Text` and `Claude3.7-Vision`. Results show that all three strategies consistently outperform both base-

Table 4: Performance with different injection strategy on *TSEvol* and *TS&Language* Benchmark (mean±std). Best results are **bolded**.

Model	Injection Strategy	Inductive	Deductive	Causal	MCQ2	**Overall**
TSLM (ChatTS-14B)	—	0.812 ± 0.007	0.597 ± 0.013	**0.732 ± 0.006**	0.590 ± 0.026	0.683 ± 0.010
GRLM (Claude3.7-Text)	—	0.763 ± 0.021	0.612 ± 0.029	0.645 ± 0.021	0.640 ± 0.014	0.665 ± 0.010
TSLM + GRLM	Intermediate	0.805 ± 0.026	0.659 ± 0.022	0.645 ± 0.010	**0.703 ± 0.037**	0.703 ± 0.006
	Late	0.791 ± 0.014	**0.667 ± 0.011**	0.703 ± 0.019	0.680 ± 0.022	0.710 ± 0.003
	Early	**0.824 ± 0.019**	0.643 ± 0.011	0.703 ± 0.019	0.690 ± 0.016	**0.715 ± 0.003**
TSLM (ChatTS-14B)	—	0.812 ± 0.007	0.597 ± 0.013	0.732 ± 0.006	0.590 ± 0.026	0.683 ± 0.010
GRLM (Claude3.7-Vision)	—	0.792 ± 0.016	0.643 ± 0.011	0.630 ± 0.009	0.690 ± 0.008	0.689 ± 0.005
TSLM + GRLM	Intermediate	0.809 ± 0.011	0.674 ± 0.000	0.663 ± 0.009	0.713 ± 0.017	0.715 ± 0.004
	Late	0.800 ± 0.019	**0.682 ± 0.011**	0.707 ± 0.009	0.697 ± 0.005	0.721 ± 0.005
	Early	**0.825 ± 0.011**	0.643 ± 0.029	**0.746 ± 0.005**	**0.730 ± 0.014**	**0.736 ± 0.002**

lines, aligned with our previous findings. Among them, *early injection* yields the strongest gains across both text and vision reasoners. One key reason is that mid/late injection requires the specialist to read and revise long reasoning traces, which lie outside the distribution of QA-style SFT and lead to drift or hallucination. Early injection aligns naturally with the specialist's strengths of producing short, focused analyses that can be directly prefixed into the reasoning trajectory.

Comparison between Prompting and Knowledge Injection. We next compare our knowledge injection approach with a prompting-based alternative. In the prompting setup, the same TSLM outputs are provided to the reasoning model as additional prompt instructions, rather than being integrated into its internal reasoning trace. Figure 4 contrasts the three strategies: baseline (zero-shot) reasoning, prompting, and injection. Across all model families, from open-source to closed-source, and across all three benchmark datasets, we observe that injection consistently outperforms prompting. This advantage arises because injection places domain knowledge directly inside the reasoning process, which encourages the model to interact with and reflect upon the knowledge more effectively. In contrast, when knowledge is only presented as external prompt instructions, the reasoning model often fails to fully incorporate it. See Appendix F for illustrative case studies.

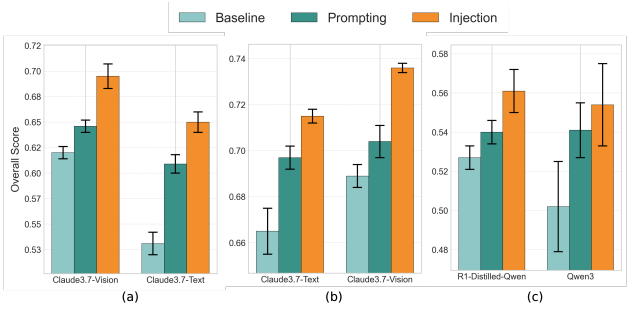


Figure 4: Comparison of baseline (zero-shot) reasoning, knowledge prompting, and knowledge injection. (a) *SenTSR-Bench* Benchmark with Qwen-VL-3B (RL) as the TSLM. (b) *TSEvol* and *TS&Language* Benchmarks with Qwen-VL-3B (RL) as the TSLM. (c) *TSEvol* and *TS&Language* Benchmarks with ChatTS-14B as the TSLM. Across all settings, the injection-based method consistently outperforms others.

4.4 Additional Analysis

We ablate the role of direct time-series access by removing the raw series from the GRLM input, showing that relying solely on the TSLM's textual summary creates an information bottleneck that limits downstream reasoning (Appendix B.1). We also compare injection against prompting-based alternatives such as few-shot, self-consistency, and tree-of-thought in terms of accuracy and inference latency (Appendix B.2). To assess training data requirements, we measure the sensitivity of TSLM performance to synthetic data diversity (Appendix B.3), and to examine whether the gains from injection can be replicated by stronger RL objectives alone, we evaluate DAPO, GSPO, and CISPO alongside GRPO (Appendix B.4). Qualitative case studies comparing standalone baselines with injection and contrasting knowledge prompting with knowledge injection are presented in Appendix F.

5 RELATED WORK

Time-series reasoning has recently attracted growing interest. One line of work studies prompting-based structured reasoning over temporal data (Jiang et al., 2025; Liu et al., 2025d; Merrill et al., 2024; Liu et al., 2025c). Another line develops specialist models post-trained on time-series–text pairs (Kong et al., 2025; Xie et al., 2024). While these approaches show promise, the former lacks domain-specific priors for

capturing key diagnostic patterns, and the latter often overfits to in-domain data and struggles with generalization. Our knowledge-injection framework aims to bridge these gaps by combining the reasoning capacity of general LLMs with domain-aligned insights from time-series specialists. Another related direction investigates interventions on the reasoning process. Prior work has explored modifying reasoning traces or internal reasoning for improved faithfulness, safety, and instruction following (Wu et al., 2025; Arcuschin et al., 2025; Baker et al., 2025) as well as methods for controlling the length of reasoning traces to balance accuracy and efficiency (Han et al., 2024a; Aggarwal and Welleck, 2025; Lee et al., 2025). Our work differs in that we explicitly inject domain knowledge from a specialized model into a general reasoning model, with a specific focus on diagnostic reasoning over time-series data. See Appendix C for additional related works on time-series reasoning in forecasting, time-series reasoning benchmarks.

6 CONCLUSION

In this paper, we introduced a *knowledge injection* framework that combines domain knowledge from time-series specialists with the strong reasoning ability of large general LLMs. We further proposed RL-based *thinking transfer* for knowledge injection, which naturally elicits analysis-first traces without supervision, enabling effective and task-aligned injection. In addition, we released *SenTSR-Bench*, a real-world benchmark for time-series diagnostic reasoning with multi-stage questions covering anomaly recognition, root-cause diagnosis, and corrective suggestions. Across *SenTSR-Bench* and public datasets, our injeciton framework achieves 7.9%–26.1% improvements over standalone baselines. We encourage exploration on *SenTSR-Bench* and further investigation of knowledge injection approaches for broader time-series diagnostic reasoning tasks.

References

Aggarwal, P. and Welleck, S. (2025). L1: Controlling how long a reasoning model thinks with reinforcement learning. *arXiv preprint arXiv:2503.04697*.

Alnegheimish, S., He, Z., Reimherr, M., Chandrayan, A., Pradhan, A., and D'Angelo, L. (2025). M2ad: Multi-sensor multi-system anomaly detection through global scoring and calibrated thresholding. In *International Conference on Artificial Intelligence and Statistics*, pages 4384–4392. PMLR.

Anthropic (2025). Claude 3.7 sonnet system card. System card, Anthropic PBC.

Arcuschin, I., Janiak, J., Krzyzanowski, R., Rajamanoharan, S., Nanda, N., and Conmy, A. (2025). Chain-of-thought reasoning in the wild is not always faithful. *arXiv preprint arXiv:2503.08679*.

Bai, S., Chen, K., Liu, X., Wang, J., Ge, W., Song, S., Dang, K., Wang, P., Wang, S., Tang, J., Zhong, H., Zhu, Y., Yang, M., Li, Z., Wan, J., Wang, P., Ding, W., Fu, Z., Xu, Y., Ye, J., Zhang, X., Xie, T., Cheng, Z., Zhang, H., Yang, Z., Xu, H., and Lin, J. (2025). Qwen2.5-vl technical report.

Baker, B., Huizinga, J., Gao, L., Dou, Z., Guan, M. Y., Madry, A., Zaremba, W., Pachocki, J., and Farhi, D. (2025). Monitoring reasoning models for misbehavior and the risks of promoting obfuscation. *arXiv preprint arXiv:2503.11926*.

Cao, Y., Fallahi, F., Dandu, M. M. K., Morishetti, L., Zhao, K., Ma, L., Subramaniam, S., Xu, J., Korpeoglu, E., Nag, K., et al. (2026). Is more context always better? examining llm reasoning capability for time interval prediction. *arXiv preprint arXiv:2601.10132*.

Chen, A., Li, A., Gong, B., Jiang, B., Fei, B., Yang, B., Shan, B., Yu, C., Wang, C., Zhu, C., et al. (2025a). Minimax-m1: Scaling test-time compute efficiently with lightning attention. *arXiv preprint arXiv:2506.13585*.

Chen, J., Feng, A., Zhao, Z., Garza, J., Nurbek, G., Qin, C., Maatouk, A., Tassiulas, L., Gao, Y., and Ying, R. (2025b). Mtbench: A multimodal time series benchmark for temporal reasoning and question answering. *arXiv preprint arXiv:2503.16858*.

Chen, M., Cui, D., Haick, H., and Tang, N. (2024a). Artificial intelligence-based medical sensors for healthcare system. *Advanced Sensor Research*, 3(3):2300009.

Chen, W., Hao, X., Wu, Y., and Liang, Y. (2024b). Terra: A multimodal spatio-temporal dataset spanning the earth. *Advances in Neural Information Processing Systems*, 37:66329–66356.

Guo, D., Yang, D., Zhang, H., Song, J., Wang, P., Zhu, Q., Xu, R., Zhang, R., Ma, S., Bi, X., et al. (2025). Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638.

Han, T., Wang, Z., Fang, C., Zhao, S., Ma, S., and Chen, Z. (2024a). Token-budget-aware llm reasoning. *arXiv preprint arXiv:2412.18547*.

Han, X., Zhang, Z., Wu, Y., Zhang, X., and Wu, Z. (2024b). Event traffic forecasting with sparse multimodal data. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pages 8855–8864.

Jaech, A., Kalai, A., Lerer, A., Richardson, A., El-Kishky, A., Low, A., Helyar, A., Madry, A., Beutel,

A., Carney, A., et al. (2024). Openai o1 system card. *arXiv preprint arXiv:2412.16720*.

Jiang, Y., Yu, W., Lee, G., Song, D., Shin, K., Cheng, W., Liu, Y., and Chen, H. (2025). Explainable multi-modal time series prediction with llm-in-the-loop. *arXiv preprint arXiv:2503.01013*.

Jin, M., Wang, S., Ma, L., Chu, Z., Zhang, J. Y., Shi, X., Chen, P.-Y., Liang, Y., Li, Y.-F., Pan, S., et al. (2023). Time-llm: Time series forecasting by reprogramming large language models. *arXiv preprint arXiv:2310.01728*.

Kong, Y., Yang, Y., Hwang, Y., Du, W., Zohren, S., Wang, Z., Jin, M., and Wen, Q. (2025). Time-mqa: Time series multi-task question answering with context enhancement. *arXiv preprint arXiv:2503.01875*.

Le, H., Do, D., Nguyen, D., and Venkatesh, S. (2025). Reasoning under 1 billion: Memory-augmented reinforcement learning for large language models. *arXiv preprint arXiv:2504.02273*.

Lee, A., Che, E., and Peng, T. (2025). How well do llms compress their own chain-of-thought? a token complexity approach. *arXiv preprint arXiv:2503.01141*.

Leite, D., Andrade, E., Rativa, D., and Maciel, A. M. (2024). Fault detection and diagnosis in industry 4.0: a review on challenges and opportunities. *Sensors (Basel, Switzerland)*, 25(1):60.

Liu, C., Xu, Q., Miao, H., Yang, S., Zhang, L., Long, C., Li, Z., and Zhao, R. (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.

Liu, H., Kamarthi, H., Zhao, Z., Xu, S., Wang, S., Wen, Q., Hartvigsen, T., Wang, F., and Prakash, B. A. (2025b). How can time series analysis benefit from multiple modalities? a survey and outlook. *arXiv preprint arXiv:2503.11835*.

Liu, H., Liu, C., and Prakash, B. A. (2025c). 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.

Liu, H., Zhao, Z., Li, S., and Prakash, B. A. (2025d). Evaluating system 1 vs. 2 reasoning approaches for zero-shot time series forecasting: A benchmark and insights. *arXiv preprint arXiv:2503.01895*.

Liu, P., Guo, H., Dai, T., Li, N., Bao, J., Ren, X., Jiang, Y., and Xia, S.-T. (2025e). Calf: Aligning llms for time series forecasting via cross-modal fine-tuning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 18915–18923.

Liu, Y., Qin, G., Huang, X., Wang, J., and Long, M. (2024). Autotimes: Autoregressive time series forecasters via large language models. *Advances in Neural Information Processing Systems*, 37:122154–122184.

Merrill, M., Tan, M., Gupta, V., Hartvigsen, T., and Althoff, T. (2024). Language models still struggle to zero-shot reason about time series. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 3512–3533.

Tavakoli, M., Chandra, R., Tian, F., and Bravo, C. (2025). Multi-modal deep learning for credit rating prediction using text and numerical data streams. *Applied Soft Computing*, 171:112771.

Wan, Z., Liu, C., Wang, X., Tao, C., Shen, H., Peng, Z., Fu, J., Arcucci, R., Yao, H., and Zhang, M. (2024). Meit: Multi-modal electrocardiogram instruction tuning on large language models for report generation. *arXiv preprint arXiv:2403.04945*.

Wang, X., Feng, M., Qiu, J., Gu, J., and Zhao, J. (2024). From news to forecast: Integrating event analysis in llm-based time series forecasting with reflection. *Advances in Neural Information Processing Systems*, 37:58118–58153.

Wang, X., Wei, J., Schuurmans, D., Le, Q., Chi, E., Narang, S., Chowdhery, A., and Zhou, D. (2022). Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

Wu, T., Xiang, C., Wang, J. T., Suh, G. E., and Mittal, P. (2025). Effectively controlling reasoning models through thinking intervention. *arXiv preprint arXiv:2503.24370*.

Xie, Z., Li, Z., He, X., Xu, L., Wen, X., Zhang, T., Chen, J., Shi, R., and Pei, D. (2024). Chatts: Aligning time series with llms via synthetic data for enhanced understanding and reasoning. *arXiv preprint arXiv:2412.03104*.

Yang, A., Li, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Gao, C., Huang, C., Lv, C., et al. (2025). Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T., Cao, Y., and Narasimhan, K. (2023). Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822.

Yeo, E., Tong, Y., Niu, M., Neubig, G., and Yue, X. (2025). Demystifying long chain-of-thought reasoning in llms. *arXiv preprint arXiv:2502.03373*.

Yu, Q., Zhang, Z., Zhu, R., Yuan, Y., Zuo, X., Yue, Y., Dai, W., Fan, T., Liu, G., Liu, L., et al. (2025). Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*.

Zhang, C., Zhang, Y., Shao, Q., Feng, J., Li, B., Lv, Y., Piao, X., and Yin, B. (2024). Bjtt: A large-scale multimodal dataset for traffic prediction. *IEEE Transactions on Intelligent Transportation Systems*.

Zhang, J., Feng, L., Guo, X., Wu, Y., Dong, Y., and Xu, D. (2025). Timemaster: Training time-series multimodal llms to reason via reinforcement learning. *arXiv preprint arXiv:2506.13705*.

Zheng, C., Liu, S., Li, M., Chen, X.-H., Yu, B., Gao, C., Dang, K., Liu, Y., Men, R., Yang, A., et al. (2025). Group sequence policy optimization. *arXiv preprint arXiv:2507.18071*.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. Yes
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. Yes
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. Yes

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. Not Applicable
   (b) Complete proofs of all theoretical results. Not Applicable
   (c) Clear explanations of any assumptions. Not Applicable

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). Yes
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). Yes
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). Yes
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). Yes

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. Yes
   (b) The license information of the assets, if applicable. Yes
   (c) New assets either in the supplemental material or as a URL, if applicable. Yes
   (d) Information about consent from data providers/curators. Yes
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. Yes

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. Not Applicable
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. Not Applicable
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. Not Applicable

Supplementary Materials

A ADDITIONAL TECHNICAL DETAILS

A.1 Details of GRPO Training Objective

For completeness, we provide the explicit form of the Group Relative Policy Optimization (GRPO) objective $\mathcal{L}_{GRPO}(\theta, R(\mathbf{z}))$ (Guo et al., 2025) used in Eq. (6).

Given a training context $(\mathbf{X}, \mathbf{q})$, we first sample a group of G complete sequences

$$\{\mathbf{z}_i\}_{i=1}^{G} \sim \pi_{\theta_{\text{old}}}(\cdot \mid \mathbf{X}, \mathbf{q}),$$

where each $\mathbf{z}_i$ contains both a reasoning trace and a final answer. We then compute scalar rewards $\{r_i\}_{i=1}^{G}$ for each sampled sequence using the composite reward function $R(\mathbf{z})$.

We normalize the rewards into advantages by subtracting the group mean and dividing by the standard deviation:

$$\hat{A}_i = \frac{r_i - \mu_r}{\sigma_r}, \quad \mu_r = \frac{1}{G}\sum_{j=1}^{G} r_j, \quad \sigma_r = \sqrt{\frac{1}{G}\sum_{j=1}^{G}(r_j - \mu_r)^2 + \gamma},$$

where γ is a small constant to ensure numerical stability. Each token $z_{i,k}$ in sequence $\mathbf{z}_i$ shares the same normalized advantage $\hat{A}_i$, ensuring stable gradient updates across contexts.

We then optimize the clipped surrogate objective with KL regularization against a frozen reference model π_{ref}:

$$\mathcal{L}_{GRPO}(\theta) = \frac{1}{G}\sum_{i=1}^{G}\frac{1}{|\mathbf{z}_i|}\sum_{k=1}^{|\mathbf{z}_i|}\min\big(\rho_{i,k}\hat{A}_i,\ \text{clip}(\rho_{i,k}, 1-\epsilon, 1+\epsilon)\hat{A}_i\big) - \beta\,\text{KL}[\pi_\theta(\cdot \mid \mathbf{X}, \mathbf{q}) \parallel \pi_{\text{ref}}(\cdot \mid \mathbf{X}, \mathbf{q})],$$

where

$$\rho_{i,k} = \frac{\pi_\theta(z_{i,k} \mid z_{i,<k}, \mathbf{X}, \mathbf{q})}{\pi_{\theta_{\text{old}}}(z_{i,k} \mid z_{i,<k}, \mathbf{X}, \mathbf{q})}$$

is the token-level importance ratio, ϵ is the PPO clipping threshold, and β is the KL regularization coefficient. This objective balances three forces: (i) improving the likelihood of high-reward completions relative to the old policy, (ii) clipping updates to maintain stability, and (iii) penalizing divergence from a reference model to prevent degeneration.

A.2 Other Injection Paradigms

Beyond early insertion, the same framework can be applied for other paradigms by changing *where* we place the snippet and *how* we shape the request. For completeness, here we introduce framework of intermediate and late injection.

Intermediate Knowledge Injection The general reasoner first drafts a partial or full reasoning trace. Although the trace is token-level in our notation, in experiments we elicit a sentence-level structure by instructing the model to reason step by step, sentence by sentence, starting from observable time-series evidence and then connecting to higher-level conclusions. Formally, we apply a deterministic segmentation operator that groups tokens into sentences,

$$\hat{\mathbf{r}}^G = [\,\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_L\,], \qquad \mathbf{s}_\ell \in V^*.$$

Along with each sentence $\mathbf{s}_\ell$, we prompt the model to output a self-reported confidence score $c_\ell \in [0,1]$. In this paper we allow the reasoner to complete the draft $\hat{\mathbf{r}}^G$ and then select the sentence with the lowest confidence,

$$\ell^* \;=\; \arg\min_{\ell \in \{1,\dots,L\}} \mathsf{Conf}(\mathbf{s}_\ell).$$

Let k^* be the token index at the start of $\mathbf{s}_{\ell^*}$. We now shape an assistance query that asks the specialist to judge this specific statement against the time series and to provide evidence or a correction,

$$\tilde{\mathbf{q}} \;=\; \mathsf{Query}_{\text{assist}}\big(\mathbf{q},\, \hat{\mathbf{r}}^G_{\le k^*},\, \mathbf{s}_{\ell^*},\, \mathbf{v}_{\text{judge}}\big),$$

where $\mathbf{v}_{\text{judge}}$ instructs the specialist to verify whether $\mathbf{s}_{\ell^*}$ is supported by the time-series $\mathbf{X}$. The specialist returns knowledge

$$\mathbf{K}^T \;\sim\; \pi^T(\,\cdot \mid \mathbf{X},\, \tilde{\mathbf{q}}\,).$$

We then perform injection by rolling back to the insertion point and inserting a brief reflection cue $\mathbf{v}_{\text{reflect}}$ between the existing trace and the specialist knowledge,

$$\mathbf{r}^{\text{Inj}}_{\le k^*} \;=\; \mathsf{Inject}_{\text{assist}}\big(\hat{\mathbf{r}}^G_{<k^*},\, \mathbf{v}_{\text{reflect}},\, \mathbf{K}^T\big) \;=\; \big[\,\hat{\mathbf{r}}^G_{<k^*},\, \mathbf{v}_{\text{reflect}},\, \mathbf{K}^T\,\big].$$

The reasoner then resumes generation for $j \ge k^*$ conditioned on $\mathbf{r}^{\text{Inj}}$ and produces the final answer following Eq. (2). This design targets the least certain statement in the draft and supplies focused, time-series grounded evidence at that point.

Late Knowledge Injection. The general reasoner first produces a complete reasoning trace $\hat{\mathbf{r}}^G$. As in the intermediate setting, we elicit a sentence by sentence structure by prompting the model to enumerate observations from the time series before drawing conclusions. Recall that we segment the trace into sentences

$$\hat{\mathbf{r}}^G \;=\; [\,\mathbf{s}_1, \mathbf{s}_2, \dots, \mathbf{s}_L\,], \qquad \mathbf{s}_\ell \in V^*,$$

where each $\mathbf{s}_\ell$ states an observation or an intermediate claim about $\mathbf{X}$. In late injection there is no confidence monitoring. Instead, we submit the entire draft to the specialist for a structured critique. We shape a critique query that includes the question, the full draft, and a critique instruction,

$$\tilde{\mathbf{q}} \;=\; \mathsf{Query}_{\text{critique}}\big(\mathbf{q},\, \hat{\mathbf{r}}^G,\, \mathbf{v}_{\text{critique}}\big),$$

where $\mathbf{v}_{\text{critique}}$ asks the specialist to examine each sentence $\mathbf{s}_\ell$ against $\mathbf{X}$, indicate whether it is supported or contradicted, explain why, and if incorrect provide a corrected statement with channel and time references. The specialist returns a knowledge sequence

$$\mathbf{K}^T \;\sim\; \pi^T(\,\cdot \mid \mathbf{X},\, \tilde{\mathbf{q}}\,),$$

which we structure as a list of per sentence judgments and corrections. We then inject after the full draft by appending a brief reflection cue followed by the specialist critique,

$$\mathbf{r}^{\text{Inj}} \;=\; \mathsf{Inject}_{\text{critique}}\big(\hat{\mathbf{r}}^G,\, \mathbf{v}_{\text{reflect}},\, \mathbf{K}^T\big) \;=\; \big[\,\hat{\mathbf{r}}^G,\, \mathbf{v}_{\text{reflect}},\, \mathbf{K}^T\,\big].$$

The reasoner performs a short refinement pass that summarizes the critique, reconciles disagreements, and updates its conclusion, then generates the final answer following Eq. (2). This late insertion supplies broad, time series grounded feedback on the entire draft and encourages reflection before finalization.

B ADDITIONAL EXPERIMENT RESULTS

B.1 Ablation on Reliance on TSLM Textual Summaries

We examine whether the GRLM can rely solely on the TSLM's textual summary, or whether its own direct access to the raw time series $\mathbf{X}$ is necessary for effective reasoning. In our full design, both the TSLM and the GRLM receive $\mathbf{X}$, and the injected summary serves as auxiliary guidance rather than the only information source. This is formalized in Eq. (3) and Algorithm 1, where the GRLM conditions on $(\mathbf{X}, \mathbf{q}, \mathbf{r})$. The motivation is to avoid a potential failure mode where errors or omissions in the TSLM summary become a single point of failure for downstream reasoning.

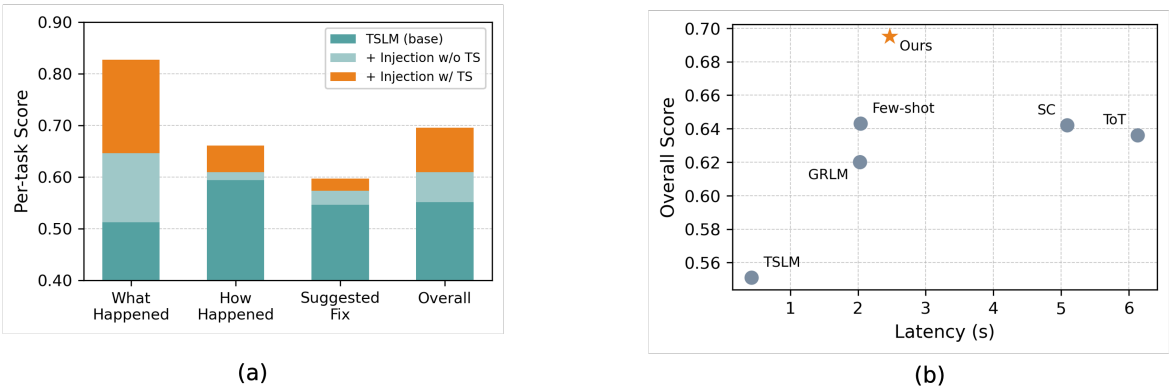


Figure 5: (a) Performance comparison between (i) the standalone TSLM, (ii) knowledge injection where the GRLM receives only the TSLM textual summary (Injection w/o TS), and (iii) full knowledge injection where the GRLM receives both the raw time series and the injected summary (Injection w/ TS). (b) Comparison of overall diagnostic accuracy versus inference latency for different methods.

To explicitly test this concern, we introduce an ablation in which the GRLM receives only the TSLM-generated textual summary, without access to the raw time series. As shown in Figure 5 (a), the "Injection w/o TS" variant improves over the standalone TSLM, indicating that transferring learned structure from the specialist is beneficial. However, it consistently underperforms the full injection setting. On average, relying only on the textual summary yields approximately a 7% improvement, whereas full injection with direct time-series access achieves around a 17% improvement. The gap is most pronounced for the "What Happened" stage, where accurate perception of temporal patterns and anomalies is critical. This ablation demonstrates that the dual-input design is essential: the GRLM does not blindly inherit the TSLM's errors, but instead combines its own perception of the time series with injected domain knowledge, leading to more robust and accurate diagnostic reasoning.

B.2 Accuracy and Latency Comparison Across Prompting-Based Alternatives and Injection.

We compare our injection-based approach against several commonly used prompting alternatives, including few-shot prompting, self-consistency (Wang et al., 2022), and tree-of-thought (Yao et al., 2023). Self-consistency is implemented with three independent reasoning runs, and tree-of-thought uses three parallel branches. As shown in Figure 5 (b), these methods consistently improve over the zero-shot GRLM baseline, confirming that structured prompting and sampling-based reasoning can enhance performance. However, all prompting-based approaches remain noticeably below the injection method in terms of final accuracy, despite incurring substantially higher inference latency. This indicates that the advantage of injection stems from transferring knowledge learned by the TSLM through training, rather than from prompt-level heuristics.

B.3 Sensitivity of TSLM performance to synthetic data diversity

To examine the sensitivity of the TSLM to the quality and diversity of synthetic training data, we conduct an ablation where the model is trained with increasing proportions of seed-generated synthetic data, ranging from no synthetic data to the full dataset. As shown in Figure 6 (a), training without synthetic data yields performance close to random guessing across all subtasks, indicating that training is essential for establishing basic time-series diagnostic reasoning ability for small models.

Once training is introduced, performance improves rapidly: using roughly 50% of the seed-generated data already recovers the majority of the final performance, while increasing diversity beyond 75% yields only marginal additional gains. This trend is consistent across subtasks, with slightly stronger saturation effects for higher-level reasoning tasks (How Happened and Suggested Fix).

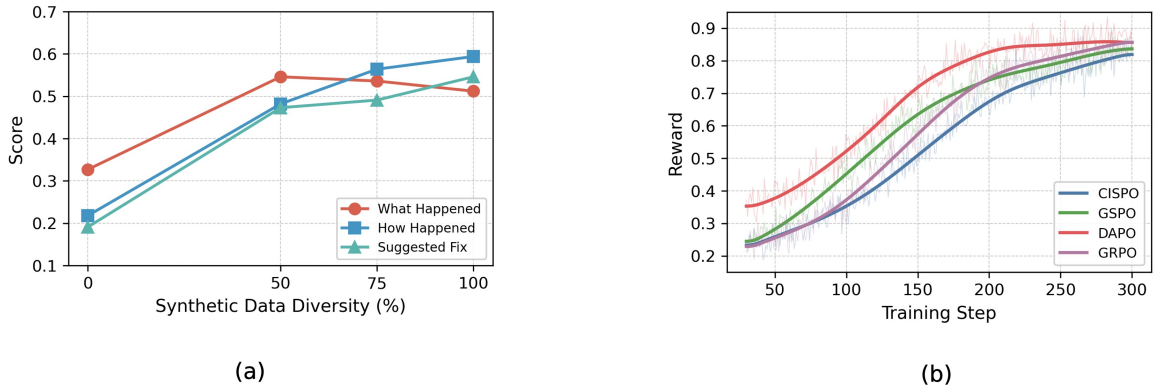


Figure 6: (a) Performance of the TSLM versus synthetic training data diversity, measured by varying the proportion of seed-generated synthetic data used during training. (b) Comparison of reward trajectories for four RL objectives (GRPO, DAPO, GSPO, CISPO) used to train the TSLM.

B.4 Reward convergence under different RL optimization methods

Motivated by recent work on efficient R1-style fine-tuning (Le et al., 2025; Yeo et al., 2025), we further examine whether more advanced RL objectives can improve TSLM training in our setting. We evaluate three representative RL objectives—DAPO (Yu et al., 2025), GSPO (Zheng et al., 2025), and CISPO (Chen et al., 2025a), alongside our GRPO baseline. As shown in Figure 6 (b), methods like DAPO yields faster and smoother reward convergence. At the same time, we observe that the final reward achieved by these methods remains similar across objectives. This suggests that, beyond convergence efficiency, overall performance is primarily constrained by the available supervision and the capacity of the model rather than the specific RL methods.

C ADDITIONAL RELATED WORK

Multi-modal Time-Series Forecasting Models Recent there are several lines of research that explores a multimodal solution for time-series analysis to incorporate the information from textual data (Liu et al., 2025b). Examples include augmenting series with domain-relevant text (Jin et al., 2023; Liu et al., 2025a, 2024, 2025e), aligning physiological signals with clinical notes (Wan et al., 2024), linking stock trends with news (Wang et al., 2024; Tavakoli et al., 2025), and incorporating geographic context (Chen et al., 2024b), traffic data (Zhang et al., 2024), or external events (Han et al., 2024b) for traffic-flow modeling. However, these work mostly focuses on forecasting tasks rather than multi-modal understanding and diagnostic tasks.

Time-Series Reasoning Models and Benchmarks. Time series reasoning has recently drawn growing interest as research moves from prediction toward explanation and diagnosis. Several works explore prompting based reasoning over temporal data (Jiang et al., 2025; Liu et al., 2025d; Merrill et al., 2024). *VLTime* (Liu et al., 2025c) represents time series as visual plots and queries multimodal models such as GPT4o for zero or few shot interpretation, while *TimeMQA* (Kong et al., 2025) formulates question answering tasks using multiple choice reasoning. Both works introduce accompanying benchmarks, *TimerBench* and *TimeMQA*, which are derived from forecasting, classification, or anomaly detection datasets rather than from diagnostic annotations. Recent datasets such as *TS&Language* (Merrill et al., 2024) and *TSEvol* (Xie et al., 2024) extend the setting to textual question–answer tasks, but their explanations are automatically generated by large language models and lack verified diagnostic grounding. Our *TSRIndustrial* benchmark differs in two key aspects: it provides human-verified diagnostic annotations and introduces a multi-stage problem structure that progresses from identifying anomalies to inferring root causes and suggesting fixes, reflecting the reasoning depth required in real-world maintenance scenarios. Concurrently, Cao et al. (2026) provide one of the first systematic investigations of LLMs on structured temporal reasoning, focusing on time interval prediction. Their findings reveal that LLMs outperform lightweight statistical baselines yet consistently underperform dedicated machine learning models, and that incorporating additional context does not always improve and can even degrade prediction quality. This formal characterization of LLM temporal reasoning capabilities lays important groundwork for the broader

time-series reasoning direction, and extending such analysis beyond interval prediction to diagnostic reasoning remains an interesting future direction.

D DATASET DETAILS

D.1 Public Dataset

We evaluate our framework on two public benchmarks, *TSEvol* and *TSandLanguage*.

TSEvol (Xie et al., 2024) consists of multiple subdatasets, among which we specifically use Dataset A, as it contains real-world time series collected from diverse domains such as AIOps, meteorology, the Numenta Anomaly Benchmark (NAB), and Oracle system metrics. The time series in Dataset A are manually annotated to mark key temporal behaviors, while the contextual prompts and root-cause options are generated automatically by LLM. The dataset includes 525 questions spanning three reasoning categories: (i) *inductive reasoning* — summarizing the physical semantics in univariate or multivariate series, (ii) *deductive reasoning* — verifying temporal conditions, and (iii) *causal reasoning* — selecting the most plausible cause under a given textual context.

TSandLanguage (MCQ2) (Merrill et al., 2024) is an open-source dataset designed for relational and comparison reasoning between two time series under textual context. The time series, questions, and answers are automatically generated by large language models. Following Xie et al. (2024), we focus on its diagnostic-style multiple-choice subset and exclude etiological reasoning and forecasting components that are not aligned with our evaluation objectives, randomly sampling 100 representative questions.

D.2 SenTSR-Bench

SenTSR-Bench is a de-identified real-world diagnostic reasoning benchmark derived from industrial sensor systems. It contains 110 multivariate time series paired with 330 human-verified diagnostic questions, spanning three progressive reasoning stages: (i) *what happened* — identifying anomalous signals and temporal patterns, (ii) *how it happened* — inferring plausible root causes behind the observed behavior, and (iii) *suggested fix* — proposing potential corrective actions. The dataset captures realistic multivariate temporal reasoning complexity, from signal interpretation to causal and prescriptive reasoning.

D.2.1 Evaluation Dataset Curation

The construction of SenTSR-Bench proceeds in three stages:

Stage 1: Signal selection and preprocessing. We start from a large pool of approximately 2,000 multivariate sensor time-series collected from real monitoring systems. From these, we identify 110 streams that display clear anomalous behaviors such as persistent deviations, sharp drops or spikes, or sudden shifts in periodicity. Each selected stream is associated with a downstream troubleshooting event in real practice, ensuring the anomalies are tied to actionable diagnostic contexts. We then apply preprocessing to standardize sampling frequency, normalize scales across sensor channels, and fully de-identify the signals by removing all system identifiers and metadata that could reveal sensitive operational information.

Stage 2: Human annotation pipeline. We develop a de-identified annotation pipeline that preserves the realism of paired textual data while protecting privacy. Human experts annotate the selected anomalous windows with concise descriptions of the observed pattern, plausible root causes, and candidate corrective actions. To prevent leakage of proprietary context, annotators are provided only with sanitized time-series segments and high-level machine categories. The resulting annotations capture domain-relevant diagnostic reasoning in natural language while guaranteeing de-identification.

Stage 3: Construction of evaluation queries. To enable systematic benchmarking, we cluster the curated time-series into families of similar anomaly types (e.g., belt failure–like patterns vs. thermal runaway patterns). From these families we generate multiple-choice questions that follow a multi-stage structure. Each query involves (i) identifying the anomalous segment, (ii) inferring its root cause, and (iii) suggesting a corrective action. Ground-truth answers are paired with distractors sampled from other clusters, ensuring that solving the task requires both correct recognition and reasoning rather than memorization.

This multi-stage curation yields *SenTSR-Bench* as a realistic and challenging benchmark, with human-authored annotations grounded in real sensor signals and a design that emphasizes both diagnostic depth and privacy protection.

D.2.2 Training Dataset Generation

Building training data at scale for diagnostic reasoning is especially challenging in the multivariate sensor setting: real-world signals are scarce, and their complexity makes direct augmentation difficult. We therefore propose a two-stage pipeline that leverages vision–language models (VLMs) to bootstrap realistic simulators from a small set of seeds.

Stage 1: Iterative code synthesis. We begin with 23 standardized and de-identified multivariate time-series, each containing channels such as vibration (acceleration, velocity) and temperature. Each seed is plotted and presented to a VLM together with high-level context prompts (e.g., "write Python code that simulates similar behavior with interpretable dynamics"). The VLM outputs candidate simulation code, which we execute to generate synthetic traces. If the output contains runtime errors or fails to reproduce core dynamics of the seed (e.g., anomaly shape, periodic structure), we refine the prompt and re-run. This iterative prompt–code–simulate cycle continues until the simulator consistently reproduces the desired behaviors. The outcome is a library of seed-aligned simulators.

Stage 2: Diversification and simplification. To scale up diversity, we prompt an LLM to transform each simulator into a stochastic generator. Deterministic heuristics are replaced with latent-state dynamics and randomized parameter draws (e.g., varying noise levels, decay rates, or event frequencies). This produces a family of realistic series rather than exact replicas. We further refactor the simulators into compact, modular forms so they can be easily reused and extended. The diversified generators collectively produce a large corpus of synthetic signals that retain the statistical and structural properties of the real seeds while covering a wider variety of operating conditions.

Finally, we apply the same query-construction pipeline as in evaluation: anomalous segments from synthetic series are paired with diagnostic labels to form QA and MCQ items. This ensures consistency between training and evaluation, while enabling large-scale supervised training from only a handful of seed signals.

E IMPLEMENTATION DETAILS

E.1 Implementation Details: Reasoning Model Baselines

We evaluate standard reasoning baselines under both zero-shot and few-shot prompting. All models are accessed through an OpenAI-compatible server implemented with `vLLM`, using HuggingFace checkpoints as backends. Unless otherwise noted, reasoning traces are obtained in a zero-shot setting, while few-shot experiments prepend a small set of curated exemplars. For few-shot prompting, for the *SenTSR-Bench* benchmark, we provide 3 randomly sampled demonstrations in an in-context learning format, inserted as prior user–assistant interactions. Each demonstration contains either the time-series image or JSON text paired with its ground-truth answer. For Qwen3, DeepSeek R1, the encoded time series far exceeds the context length. In such cases, we include only the question and answer template in the demonstrations, omitting the full time-series input.

Encoding time series for LLM input. For image encoding, we render multivariate time series as stacked line plots using `matplotlib`. Each channel is placed in a vertically aligned subplot with labeled axes and channel identifiers, following best practices for visual clarity. Detailed plotting functions are provided in the released source code.

For text encoding, we convert each channel into a structured JSON-like format. The following template illustrates the format used to render time-series data into textual tokens for inclusion in prompts:

```
{
  "Series 1": [0.25, 0.31, 0.28, ...],
  "Series 2": [1.02, 1.13, 0.95, ...],
  "Series 3": [-0.42, -0.38, -0.41, ...]
}
```

This structured form facilitates tokenization and preserves the alignment of values across channels. When column names are available, they are preserved; otherwise, generic names are assigned.

Long-context adaptation. For certain benchmarks such as *TSEvol*, multivariate time-series inputs can exceed 50k tokens when encoded as text. To accommodate these cases, we apply RoPE scaling to extend the context length of open-source models such as Qwen3 and DeepSeek R1, ensuring that the full series can be processed without truncation. This scaling is necessary for faithfully grounding reasoning in long multivariate signals.

Infrastructure. All open-source models are hosted on AWS EC2 instances equipped with 8×A100 GPUs, served through `vLLM`. Closed-source reasoning models are accessed via AWS Bedrock.

E.2 TSLM Post-training

All time-series specialists (TSLMs) are initialized from the public `Qwen-VL-3B-Instruct` checkpoint. Post-training is carried out in two stages: supervised fine-tuning (SFT) and reinforcement learning (RL) with verifiable rewards. For the public benchmarks *TSEvol* and *TS&Language*, we fine-tune on 3k causal reasoning tasks from the *TSEvol* SFT set, restricting training to causal tasks to test cross-task generalization to inductive, deductive, and MCQ-2 tasks at evaluation. For the *SenTSR-Bench* benchmark, SFT data is constructed from the curated *What happened* and *How happened* stages, leaving the *Suggested fix* stage unseen for out-of-distribution evaluation. SFT training uses a cutoff length of 4,096 tokens, per-device batch size of 4 with gradient accumulation of 2 (effective batch size 64 on 8 GPUs), learning rate of 1×10^{-5}, and cosine decay scheduling with warmup ratio 0.1.

For reinforcement learning, we adopt Group Relative Policy Optimization (GRPO) to elicit analysis-first completions without explicit thinking supervision. For *TSEvol*, RL training again focuses on causal tasks, and for *SenTSR-Bench* we apply it to the *What happened* and *How happened* datasets. RL training is configured with KL divergence coefficient $\beta = 0.001$, group size $G = 8$, maximum sequence length $L_{\max} = 512$, and PPO clipping threshold of 0.1, with an effective batch size of 16 on 8 GPUs, with a learning rate of 1×10^{-6}.

E.3 Practical Implementation of Knowledge Injection.

The injection workflow is straightforward to implement with standard LLM APIs. For models and servers that support *assistant prefill* (e.g., OpenAI-compatible endpoints), we directly seed the private trace by pre-inserting $[\,\langle\text{think}\rangle,\ \mathbf{r}^{\text{Inj}}_{\leq k}\,]$ as the assistant's initial tokens; the general reasoner π^G then continues generation conditioned on this prefix. For providers that do not expose editable thinking buffers (in some closed-source reasoning models), we use an instructional proxy: wrap $\mathbf{r}^{\text{Inj}}_{\leq k}$ inside the models's recommended "thinking template" tags in the user/system message (e.g., a documented `<thinking>...</thinking>` block) and instruct the model to begin its thinking process with the instructed template. In practice this proxy reliably steers the internal reasoning trace and reproduces the effect of in-chain injection.

E.4 Prompt Design for Injection Strategies

We provide the prompt templates used in experiments for evaluating different *knowledge injection positions*. These prompts are designed for a strong instruction-following TSLM (`ChatTS-14B`) paired with general reasoning LLMs (GRLMs). The goal is to examine how injecting time-series knowledge at different points in the reasoning process, including *early*, *intermediate*, and *late* injection, affects overall reasoning performance.

Early Injection. In the early injection setup, the TSLM first produces structured, quantitative observations from the time series, which are inserted at the start of the GRLM's reasoning process to guide the subsequent chain of thought.

TSLM (Observation Generation)

```
You are analyzing a time series to extract key quantitative observations that
help answer the question.  Provide detailed, objective numerical observations
by following these guidelines:  1.  Make numbered, precise observations about
the quantitative aspects of the time series.  2.  Be specific about values,
```

```
positions, and magnitudes when describing features.  3.  Begin each observation with
"Observation 1:", "Observation 2:", etc.
Start your response with:  "To answer this question, I need to carefully analyze the
time series.  Here are my observations:  Observation 1...  Observation 2..."
```

GRLM (Reasoning with Early Injection)

```
<think> [TSLM Observations] Wait, let me summarize and reflect on the previous
observations from the time series, and then continue my reasoning process to derive
the final answer...
```

Intermediate Injection. In this setting, the GRLM begins reasoning but calls for the TSLM's input when encountering uncertainty (identified as a low-confidence step). The TSLM then provides clarifications, which are integrated back into the GRLM's ongoing reasoning.

TSLM (Intermediate Feedback)

```
You are assisting with a time series reasoning process.  Here is the question and
the current partial reasoning:  [Partial Thought / Low-Confidence Segment] Please
analyze whether this reasoning point is correct based on the time series, why or
why not, and how it relates to answering the question.  Be specific about numerical
values and time positions.
```

GRLM (Reasoning with Intermediate Injection)

```
<think> [Partial Reasoning] I am uncertain about this point:  [Low-Confidence
Segment].  Let me reconsider it with the following clarification from the
time-series model:  "[TSLM Feedback]" ...
```

Late Injection. Under late injection, the GRLM first completes its reasoning trace. The TSLM then reviews the reasoning for factual consistency with the time series, and the GRLM revises its conclusion accordingly.

TSLM (Late Review)

```
You are reviewing a completed reasoning process to check whether its quantitative
claims about the time series are correct.  For each observation, discuss whether it
is correct or incorrect.  If incorrect, provide the accurate interpretation of what
the data shows.  Example:  "For Observation 1, after review, it is incorrect.  In
fact...  For Observation 2..."
```

GRLM (Revision with Late Injection)

```
<think> [Original Reasoning Trace] Wait, let me reexamine my previous reasoning
based on the review below:  [TSLM Review] ...
```

Additional Adaptations. For the **RL-honed TSLM**, we apply *early injection* by directly using the model's self-generated reasoning trace from R1-style GRPO training as the injected knowledge, without explicit prompting; for **closed-source models** (e.g., `Claude-3.7`), injected content is wrapped in `<thinking>...</thinking>` delimiters, followed by an instruction such as "continue the thinking process above."; for the **prompting-based baseline**, the same TSLM-generated content is provided externally as additional context:

Prompting Baseline

```
Here is an analysis from a time-series model that is good at time-series analysis
but may be limited in general reasoning:  [TSLM Observations].
```

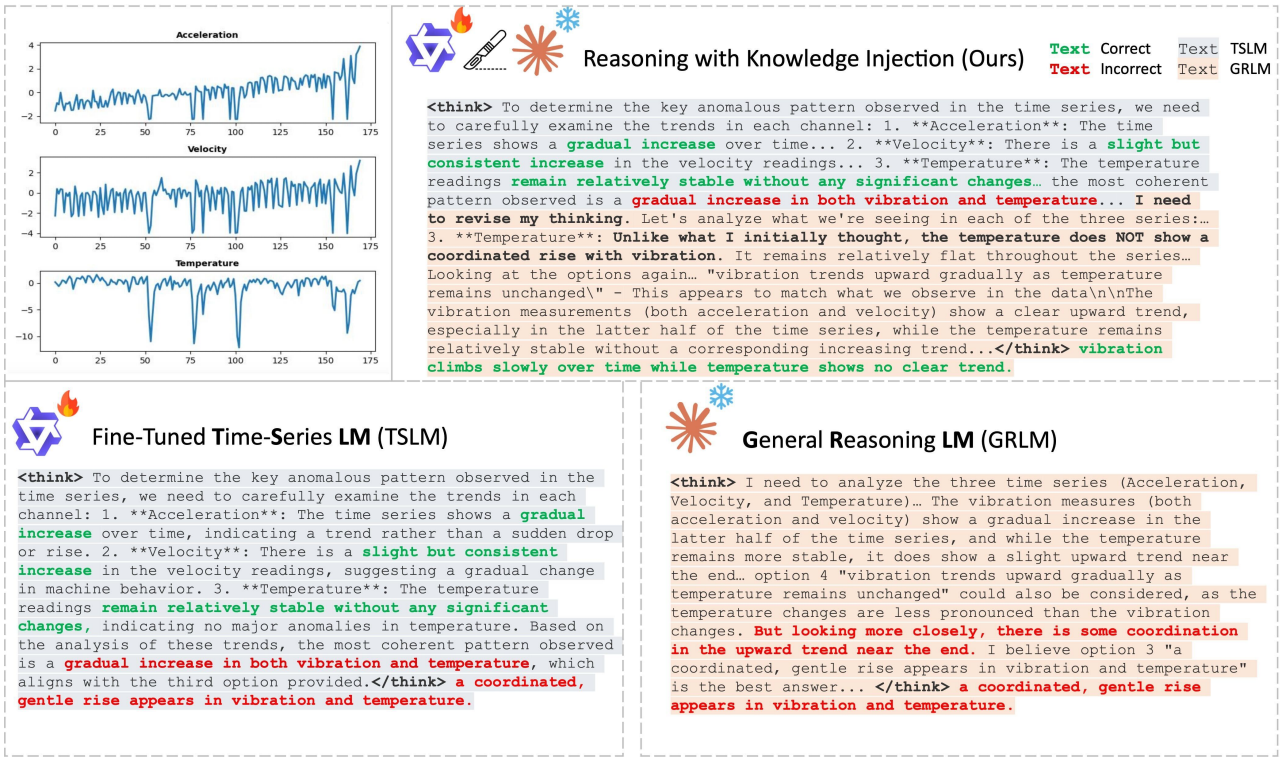
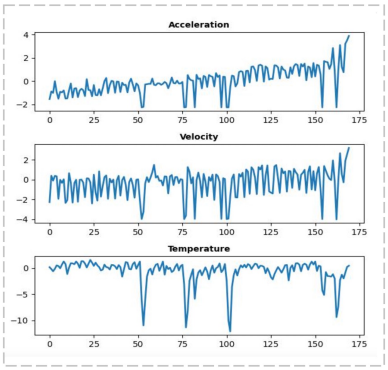


Figure 7: Case study on *knowledge injection* versus standalone baselines. The **TSLM** correctly detects rising vibration and stable temperature but hallucinates a joint increase in both, yielding an incorrect diagnosis. The **GRLM** similarly misreads the series, assuming a late temperature rise. Our method leverages the TSLM's accurate signal interpretation while correcting its reasoning flaw, producing the correct final diagnosis.

F ADDITIONAL CASE STUDY

We present two qualitative case studies that illustrate the benefits of our knowledge injection framework. Figure 7 compares the standalone TSLM, the standalone GRLM, and our injection-based method on a diagnostic reasoning example. The TSLM correctly detects rising vibration and stable temperature but hallucinates a joint increase in both signals, yielding an incorrect diagnosis. The GRLM similarly misreads the series, assuming a late temperature rise. By injecting the TSLM's accurate signal-level observations into the GRLM's reasoning trace, our method corrects the reasoning flaw while preserving domain-grounded pattern recognition, producing the correct final diagnosis. Figure 8 further contrasts knowledge *prompting* with knowledge *injection*. When the TSLM analysis is provided as an external prompt, the GRLM reasons largely in isolation, leading to insufficient use of domain knowledge. In contrast, the injection-based approach integrates the TSLM's analysis directly into the reasoning flow, enabling joint exploration and progressive narrowing of hypotheses, and resulting in a correct diagnosis.



Reasoning with Knowledge Injection (Ours)

Text Correct Text TSLM
Text Incorrect Text GRLM

<think> To identify the best corrective action, we need to carefully analyze the trends in the vibration (acceleration and velocity) and temperature time series... The acceleration and velocity series show a steady increase over time... The best corrective action for the event implied by the anomalous pattern in these time series is to **continue monitoring gradual imbalance development and schedule balancing service.** Let me analyze the time series data shown in the image... Based on this analysis, **I agree with the previous thinking process.** The gradual increase in vibration amplitude (both acceleration and velocity) without corresponding temperature increases suggests mechanical imbalance developing over time. This is best addressed by monitoring the imbalance development and scheduling balancing service, rather than immediate intervention or focusing on lubrication or belt issues... **</think> continue monitoring gradual imbalance development and schedule balancing service.**

Reasoning with Knowledge Prompting

<prompt>...Here is an analysis from a time series model: To identify the best corrective action, we need to carefully analyze the trends in the vibration (acceleration and velocity) and temperature time series... The acceleration and velocity series show a steady increase over time... The best corrective action for the event implied by the anomalous pattern in these time series is to **continue monitoring gradual imbalance development and schedule balancing service. <prompt> <think>** I'm looking at the three time series plots provided for acceleration, velocity, and temperature of a machine monitoring system... Given the data patterns I'm seeing, I think the best answer is: \"Check belt tracking immediately for fresh rub marks and remove contact points\". This would address both the increasing vibration levels and the periodic disruptions we're seeing across all parameters. **</think> Check belt tracking immediately for fresh rub marks and remove contact points.**

Figure 8: Case study on knowledge prompting versus knowledge injection. In the prompting-based approach, the **GRLM** reasons largely in isolation, leading to insufficient use of domain knowledge. In contrast, the injection-based approach integrates the **TSLM**'s discussion directly into the reasoning flow, enabling joint exploration and narrowing of hypotheses, and resulting in a correct diagnosis.