ZERO-SHOT TIME-SERIES QUESTION ANSWERING VIA DECOUPLED PERCEPTION AND REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Time-series question answering (TSQA) requires grounding linguistic queries and diverse answer formats in complex numerical observations. However, existing methods heavily overfit to specific datasets and struggle to generalize when input series, question contexts, and answer requirements shift simultaneously. To address this challenge, we propose **TSHarness**, an agentic framework that establishes a decoupled workflow for cross-dataset zero-shot TSQA. At its core, TSHarness divides and conquers numerical perception and contextual reasoning via a structured *Time-Series Perception State* (TPS). Guided by a reusable memory of analytical knowledge, a learned *Tool Selector* adaptively invokes numerical tools to extract salient statistical and temporal features into the TPS. The answering agent then performs semantic reasoning over the TPS to generate target outputs, triggering iterative re-perception through the feedback loop when evidence is deemed insufficient. By separating numerical feature extraction from question-specific reasoning, TSHarness eliminates the need for target-side training or answer feedback, providing a generalizable, cost-efficient foundation for zero-shot TSQA.

1 INTRODUCTION

Time-series question answering (TSQA) turns numerical observations into answers about temporal patterns, events, predictions, and decisions (Kong et al., 2025; Yu et al., 2026; Guan et al., 2026). Current TSQA methods develop useful temporal capabilities through numerical-text alignment and task-specific training (Wang et al., 2025a; Kong et al., 2025). Yet, we observe that performance gains of these models on one dataset do not reliably transfer to another (Figure 1(a)), where perception, reasoning, and answer production must be recombined.

We attribute this transfer gap to prevailing paradigms that treat TSQA as a monolithic task, tightly entangling numerical perception with dataset-specific answer generation. Consequently, such end-to-end models underperform in *zero-shot* TSQA tasks under two entangled distribution shifts: *perceptual shifts in input series* across sampling rates, sequence lengths, missingness, and channel dimensions; and (2) *contextual shifts in task specifications* where question semantics and target answer requirements diverge together, spanning multiple-choice, true/false, and ordered labels under heterogeneous evaluation rubrics. These concurrent shifts fundamentally impede knowledge reuse across domains: fine-tuned parameters of learning-based methods overfit to source dataset correlations (*parameter reuse failure*), while stored analytical procedures and memories in agent-based methods fail to generalize across divergent benchmarks (*analytical reuse failure*).

To overcome this dual transfer bottleneck, our core design principle is to decouple continuous numerical perception from context-specific reasoning. We investigate this paradigm under the rigorous setting of cross-dataset zero-shot transfer, where a source-developed system directly answers target questions without any target-side fine-tuning or answer feedback. Accordingly, we present TSHARNESS, an agentic system centered on an intermediate representation termed the Time-Series Perception State (TPS), which explicitly isolates raw numerical observations from task-dependent answer synthesis. Within this framework (Figure 1(b)), a frozen backbone LLM instantiates two collaborative roles: a PERCEIVER that extracts quantitative temporal properties into the TPS, and an ANSWERER that conditions on the TPS to resolve the contextual question.

Specifically, the TPS establishes a unified analytical basis through standardized fields for quantitative temporal primitives, alongside flexible extensions for question-specific analyses. During execution,

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

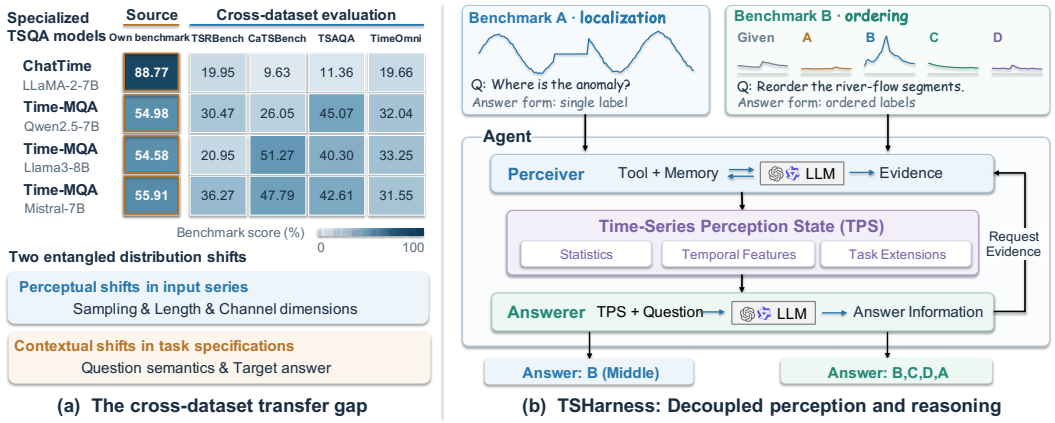

Figure 1: **Motivation and overview of TSHARNESS.** (a) Specialized TSQA models exhibit uneven cross-dataset performance as time-series inputs, question context, and answer requirements change. (b) TSHARNESS decouples perception from answering through a unified TPS representation.

guided by reusable procedural memory, the PERCEIVER dynamically invokes numerical tools to populate the TPS, translating raw series into domain-agnostic facts. The ANSWERER then reasons over the accumulated TPS and question context to synthesize the final output. Crucially, when existing evidence is deemed insufficient, the ANSWERER initiates an iterative feedback loop back to the PERCEIVER, requesting targeted, fine-grained measurements. Furthermore, to mitigate redundant computational overhead, a lightweight reinforcement-learning (RL) TOOL SELECTOR adaptively prunes sub-optimal tool proposals by balancing answer correctness against execution latency on source tasks. Through this decoupled architecture, source-developed analytical capabilities are dynamically recombined at runtime, enabling robust zero-shot generalization across unseen benchmarks while keeping the underlying backbone and analytical resources entirely fixed.

Our primary contributions are summarized as follows:

- We systematically formulate the challenge of zero-shot TSQA, identifying that coupled perceptual and contextual shifts trigger distinct parameter and procedural reuse failures in existing methods.
- We propose TSHARNESS, which isolates numerical perception from contextual reasoning via the intermediate TPS. Guided by reusable memory, collaborative PERCEIVER and ANSWERER agents resolve queries through an active evidence loop.
- We develop a lightweight RL-based approach to balance answer correctness against execution cost, substantially pruning redundant tool calls while maintaining competitive accuracy.
- We benchmark zero-shot transfer of TSHARNESS from a single source to five heterogeneous target datasets. It remarkably outperforms same-backbone baselines without any target-side adaptation.

2 RELATED WORK

Language-based forecasting (Gruver et al., 2023; Jin et al., 2024; Liu et al., 2024) and pretrained forecasters (Ansari et al., 2024; Woo et al., 2024) support transfer of numerical prediction capabilities. TSQA broadens this scope to temporal reasoning (Zhou et al., 2026; Jing et al., 2026; Yu et al., 2026; Gwiazda et al., 2026) and contextual sensor interpretation (Reichman et al., 2025). Recent benchmarks reveal cross-domain weaknesses in time-series-adapted LLMs (Yin et al., 2026) and distinguish scale selection, localization, and cross-interval integration (Han et al., 2026). Learned approaches connect numerical observations to language through expanded vocabularies in ChatTime (Wang et al., 2025a), contextual QA training in Time-MQA (Kong et al., 2025), and synthetic alignment data in ChatTS (Xie et al., 2025); TimeOmni-1 further develops reasoning through reward-guided training (Guan et al., 2026). TSHARNESS makes statistical and temporal properties explicit through the TPS, separating their representation from question-specific interpretation. This provides a common analytical basis for applying a system constructed on one dataset to questions and answer requirements in others.

Chain-of-thought prompting structures intermediate reasoning (Wei et al., 2022); tool-augmented agents connect reasoning to execution (Yao et al., 2023; Gao et al., 2023; Schick et al., 2023) and

reuse feedback, skills, and memory (Shinn et al., 2023; Wang et al., 2024; 2025b; Xu et al., 2025). Planning–execution separation (Xu et al., 2023), parallel scheduling (Kim et al., 2024), and modular architecture search (Shang et al., 2025) improve orchestration, while reward-based training develops LLM tool-use policies (Qian et al., 2025; Feng et al., 2026a). Time-series frameworks specialize these capabilities: TS-Agent combines numerical tools with evidence gathering and verification (Liu et al., 2026), AION organizes temporally grounded workflows and review (Zhan et al., 2026), and KairosAgent learns reasoning–forecasting interactions from forecasting feedback (Feng et al., 2026b). Our closest harness baseline, TimeClaw, integrates temporal tools, capability evolution, and episodic retrieval (Li et al., 2026). TSHARNESS focuses on the structure of evidence passed between perception and answering. TPS gives measured series properties a stable field structure and records task-specific calculations in separate extensions. This division lets the Answerer adapt its interpretation and evidence requests while retaining a common interface to numerical perception. A complementary RL selector learns to refine base-tool subsets with the backbone fixed, balancing answer correctness against tool executions.

3 METHOD

This section provides the technical details of TSHARNESS. Sec. 3.1 introduces our problem formulation and decoupled execution pipeline for cross-dataset zero-shot TSQA. Sec. 3.2 and Sec. 3.3 detail our tool-augmented numerical perception and establish the Time-Series Perception State (TPS), which isolates objective temporal properties from task-specific reasoning. Sec. 3.4 establishes an active harness loop for iterative evidence reasoning. Sec. 3.5 presents an RL-based TOOL SELECTOR that balances answer correctness against tool execution cost.

3.1 FRAMEWORK OVERVIEW

TSHARNESS builds on a foundational insight: while questions across heterogeneous TSQA benchmarks differ drastically in textual framing and answer specifications, they fundamentally rely on a shared set of underlying statistical and temporal properties. Formally, let a TSQA query instance be denoted as $u = (X, c, \kappa)$, where $X = (X_1, \ldots, X_m)$ comprises the observed numerical series along with their metadata descriptions, c denotes the question text and relevant background context, and κ encapsulates the target answer specifications (*e.g.*, categorical options, numerical tolerance, or formatting rubrics). Let Z_r denote the TPS representation after r accepted evidence refinement updates. The overall execution workflow of TSHARNESS is formulated as:

$$Z_0 = \mathrm{Init}(u), \qquad Z_{r+1} = \mathrm{Update}(Z_r, q_r; X), \qquad \hat{y} = \mathrm{Answer}(Z_R, c, \kappa). \tag{1}$$

where $\mathrm{Init}(\cdot)$ constructs Z_0 from base measurements and question-specific analyses under the PERCEIVER's validated plan, guided by procedural memory and optionally refined by the TOOL SELECTOR. At iteration r, an evidence request q_r dispatched by the ANSWERER invokes specific numerical operations, appending newly verified empirical findings to update the state from Z_r to Z_{r+1}. The loop terminates at step R when the accumulated evidence is deemed sufficient or the execution budget is reached. $\mathrm{Answer}(\cdot)$ then synthesizes the final prediction $\hat{y}$ conforming to κ. A single frozen backbone LLM supports planning and answering through role-specific calls, coordinated by the harness loop in Figure 2.

We construct the system exclusively on a single source dataset, using execution trajectories and ground-truth answer feedback to curate the tool library, analytical memory bank, and policy parameters of the TOOL SELECTOR. During cross-dataset zero-shot transfer, all source-developed components, prompting templates, and model parameters remain strictly frozen. The standardized tool interfaces, the TPS schema, and procedural heuristics transfer seamlessly across benchmarks; numerical evidence is computed entirely on-the-fly for each target instance. Target tasks steer the analytical process strictly via dynamic tool planning and runtime evidence requests, completely eliminating the need for target-side parameter training or reference-answer feedback. Appendix A specifies more details of the construction and evaluation partitions.

3.2 PERCEPTION AND TOOL ORCHESTRATION

The perception phase grounds qualitative queries into verifiable numerical measurements via a suite of callable computational tools. Each tool τ declares its supported analyses, input requirements, and

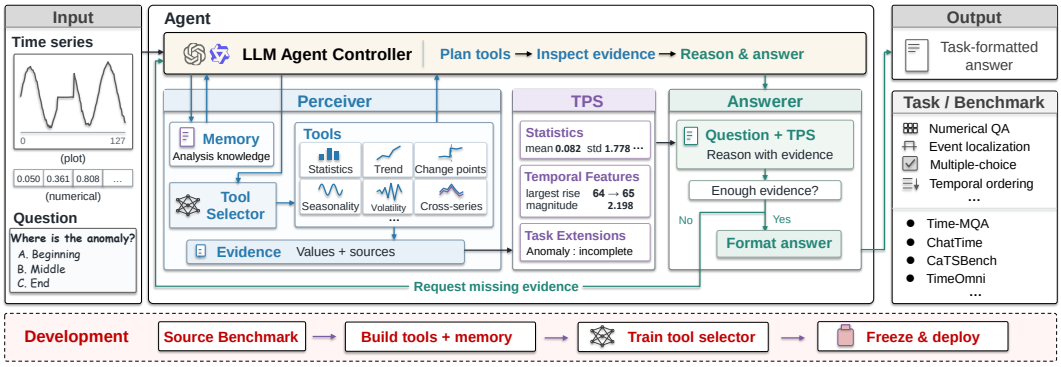

Figure 2: **Workflow of TSHARNESS.** Guided by procedural memory, the PERCEIVER populates the TPS with grounded measurements, refined by the TOOL SELECTOR. The ANSWERER interprets the accumulated evidence alongside the question, actively initiating an iterative feedback loop to request further evidence. Source-developed resources are completely frozen for zero-shot transfer.

output fields. These declarations define the eligible tools $\mathcal{T}(X_j)$ for a given series X_j. Eligibility checks account for sequence length and channel availability, while arguments specify the channel or interval to analyze. Conditioned on a bounded view of the series, the question, and tool descriptions, the PERCEIVER proposes an initial execution plan:

$$B_j = \big[(\tau_{jk}, \alpha_{jk})\big]_{k=1}^{K_j}, \qquad \tau_{jk} \in \mathcal{T}(X_j),$$
$$\mathcal{E}_j = \bigcup_{(\tau,\alpha)\in\widetilde{B}_j} \mathrm{Exec}(\tau, X_j; \alpha). \tag{2}$$

where (τ_{jk}, α_{jk}) pairs an eligible tool with its parameterized arguments. Memory provides procedural guidance through entry summaries and a retrieval query generated with the plan. This proposed sequence is validated and optionally refined by the TOOL SELECTOR (Sec. 3.5) to yield the executable plan $\widetilde{B}_j$. Executing $\widetilde{B}_j$ against the full-resolution observations X_j returns structured records $\mathcal{E}_j$, where each entry contains field names, values, source tools, and unique evidence identifiers. A question about a spike or level shift can pair value-profile measurements of abrupt changes with change-point localization, providing complementary evidence for TPS construction.

The standardized execution interface accommodates tool combinations and arguments tailored to different series characteristics. Tools operate directly on the original numerical arrays, preserving fine-grained temporal indices that can be lost in text serialization. The extracted fields retain their analytical meaning across domains, giving the ANSWERER a consistent basis for interpreting the evidence. Appendix B details tool families, applicability checks, and an illustrative execution.

3.3 TIME-SERIES PERCEPTION STATE

The TPS defines a standardized interface for reading measured series properties and adding question-specific analyses. A stable core organizes statistical and temporal fields, while named extensions retain the scope and results of additional calculations. Its state Z_r is defined by

$$Z_r = (\mathcal{C}, \mathcal{D}_r, \mathcal{Q}), \qquad \mathcal{C} = \big\{j \mapsto (\iota_j, \mathcal{S}_j, \mathcal{F}_j, h_j)\big\}_{j=1}^{m}, \tag{3}$$

where $\mathcal{C}$ represents the static core perception, $\mathcal{D}_r$ denotes dynamic question-specific extensions, and $\mathcal{Q}$ denotes the set of required evidence types. The core $\mathcal{C}$ provides a reusable description of the series. The question determines which measurements are acquired, while their meaning remains defined by the numerical operations. The dynamic extensions $\mathcal{D}_r$ capture analyses whose setup depends on the current task, including comparisons among supplied candidates and combinations of measured properties. The ANSWERER interprets both under the question context and answer requirements, retaining the core as task-specific analysis expands. The core is fixed within a question, while its measured values are recomputed for each new input series. The same field definitions therefore apply to newly measured series across tasks.

Fixed rules derive $\mathcal{Q}$ from question text, task descriptions, and public task specifications for coverage checks during answering. An evidence type identifies a required field or analytical result, such as a

trend estimate or a comparison between candidate series. These requirements remain fixed for the question while the harness acquires the corresponding evidence.

Evidence grounding. Within the core $\mathcal{C}$, ι_j stores metadata (series identities, channel configurations, and valid temporal spans); $\mathcal{S}_j$ and $\mathcal{F}_j$ record statistical summaries (*e.g.*, mean, variance, quantile bounds) and temporal primitives (*e.g.*, trend monotonicity, dominant frequencies, prominent changepoints), respectively; and h_j is an LLM-generated summary. The PERCEIVER selects which measured fields to retain and generates the summary h_j. The runtime copies field values directly from the corresponding tool records in $\mathcal{E}_j$, checks their supporting references, and preserves links to their sources. Unmeasured fields remain unavailable. The resulting core preserves measured properties and their provenance for interpretation.

Each analytical extension has the form $\mathcal{D}_r[\nu] = (q, V, I, \delta)$, linking an analysis request q to returned evidence V, supporting identifiers I, and a coverage status $\delta \in \{\text{complete}, \text{incomplete}\}$. An LLM planning call uses the question, evidence requirements, and core findings to populate $\mathcal{D}_0$ before answering. During answering, further requests address needs that emerge when interpreting this evidence. Both stages use the same extension structure and available tools, keeping question-specific analysis compatible with the core state. In the level-shift case (Appendix B.5), the core records a -2.48 drop at index 32; an extension gathers anomaly and change-point evidence to characterize it.

3.4 ANSWERING WITH THE HARNESS LOOP

The ANSWERER evaluates the tuple (c, κ, Z_r) to determine whether to emit a terminal response or initiate a targeted evidence acquisition step in Eq. (1). Before evidence-conditioned answering, the same backbone generates an initial hypothesis from the series and question without TPS evidence. This provisional answer is stored in a named extension, separately from the measured properties. The ANSWERER reassesses it against the accumulated evidence; Appendix B details the preliminary call. The iterative loop unfolds through three consecutive phases:

1. **Gap inspection.** The runtime exposes missing evidence types:
$$\mathcal{M}_r = \mathcal{Q} \setminus \mathrm{Cov}(Z_r), \tag{4}$$
where Cov denotes evidence types supplied by the available TPS fields and analytical results. Coverage tracks whether the required evidence is present; the ANSWERER assesses whether it resolves the question or motivates further analysis.
2. **Targeted re-perception.** A request $q_r = (J_r, O_r, A_r, \mathcal{Q}_r)$ specifies target series indices, operations, arguments, and required evidence. After validation and execution, the evidence bundle V_r, identifiers I_r, and status δ_r are appended under a fresh name ν_r:
$$\mathcal{D}_{r+1} = \mathcal{D}_r \cup \{\nu_r \mapsto (q_r, V_r, I_r, \delta_r)\}, \qquad Z_{r+1} = (\mathcal{C}, \mathcal{D}_{r+1}, \mathcal{Q}). \tag{5}$$
Unsupported, duplicate, or out-of-budget requests are rejected.
3. **Constrained answer synthesis.** The ANSWERER can commit a response at any iteration and proceeds with the available state when the request budget is exhausted. The output is parsed according to κ, with an overall step limit bounding execution.

In the level-shift case, the abrupt drop identifies where the series changes, while the persistence of the new level distinguishes a level shift from an isolated spike. The ANSWERER maps this measured pattern to the requested answer option. The measurements describe the series; (c, κ) determines their interpretation and answer form. Further requests extend $\mathcal{D}_r$ while retaining $\mathcal{C}$, allowing contextual reasoning to continue through the same numerical perception interface.

3.5 RL-BASED TOOL SELECTION

While dynamic tool invocation ensures expressive perception, executing all candidate tools within the initial tool plan incurs substantial latency and computational overhead. We train a lightweight TOOL SELECTOR with final-answer feedback to refine that plan. Tool selection is framed as a Markov Decision Process $(\mathcal{S}, \mathcal{A}, P, r, H)$. An episode spans a complete source question. At decision step t,
$$s_t = (u, z_t, \mathcal{T}_t, B_t), \qquad a_t \in \mathcal{A}(s_t) = \{A \subseteq \mathcal{T}_t : A \neq \varnothing\}, \tag{6}$$
where u is the question input defined in Sec. 3.1, z_t records the current series, execution history, and remaining analysis budget, $\mathcal{T}_t$ contains the eligible tools, and $B_t$is the base proposal from the

PERCEIVER. Each action selects a tool subset for execution. Transition P executes the selected tools, populates the harness state, and progresses either to subsequent series or to the final answering stage within a bounded horizon H. The policy uses the PERCEIVER's proposal B_t as an informed prior for tool selection. For candidate tool $i \in \mathcal{T}_t$, we define a prior inclusion probability $p^0_{t,i}$ ($p^0_{t,i} > 0.5$ if proposed in B_t, and a base rate otherwise). A lightweight scoring network f_θ maps a dense feature vector $\phi_{t,i} = \phi(s_t, i)$ (encoding series characteristics, tool categories, and query semantics) to an additive logit shift:

$$p_{t,i} = \sigma\big(\mathrm{logit}(p^0_{t,i}) + f_\theta(\phi_{t,i})\big)\,,$$

where σ is the sigmoid function. Tool inclusion decisions follow a joint Bernoulli policy conditioned on selecting at least one operation; Appendix C gives its normalized probability in Eq. (11). Initializing f_θ near zero preserves the PERCEIVER's initial proposal prior, allowing stable policy exploration. The default deployment rule thresholds $p_{t,i} \geq 0.5$ deterministically (defaulting to $\arg\max_i p_{t,i}$ if no tool exceeds the threshold). Conservative deployment moderates the learned adjustment and retains additional evidence through the constraints in Appendix C.3.

The policy is optimized to maximize answer correctness while minimizing superfluous tool calls:

$$R_{\mathrm{tool}}(u) = \mathbf{1}[\hat{y} = y] - \lambda_{\mathrm{tool}} N_{\mathrm{base}}(u), \tag{7}$$

where $\hat{y}$ is the predicted answer, y is the source reference answer, and $N_{\mathrm{base}}(u)$ counts executions governed by the selector across the question. The coefficient $\lambda_{\mathrm{tool}} > 0$ sets the execution penalty, using tool count as a cost proxy for numerical analysis overhead. We optimize f_θ using Proximal Policy Optimization (PPO) (Schulman et al., 2017). The policy gradient is computed over multiple trajectory rollouts per source instance, with the baseline defined as the leave-one-out rollout mean to reduce variance. Once trained on source tasks, the parameters θ of the TOOL SELECTOR are frozen alongside the backbone LLM, delivering an adaptive, cost-effective perception frontend across all zero-shot target evaluations.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

We evaluate on TSQA tasks from six benchmarks: TSRBench (Yu et al., 2026), Time-MQA (Kong et al., 2025), ChatTime (Wang et al., 2025a), CaTSBench (Zhou et al., 2026), TSAQA (Jing et al., 2026), and TimeOmni-1 (Guan et al., 2026). These benchmarks span multiple question types and assess a broad range of time-series understanding and reasoning capabilities, with diverse time-series inputs, question contexts, and answer requirements. TSRBench serves as the source benchmark for TSHARNESS construction, while the other five provide targets for cross-dataset evaluation. Appendix A provides further details.

We evaluate our approach in two settings: • *In-distribution (ID) evaluation.* We use TSRBench to develop the harness, including constructing numerical tools and organizing analysis knowledge into Memory. Following prior work (Li et al., 2026), we use the 10% TSRBench test split for evaluation. • *Zero-shot evaluation.* We directly deploy the TSRBench-developed harness on the five target benchmarks under a zero-shot protocol. For comparisons among agentic methods, we randomly sample a fixed subset of 412 QA examples per target benchmark, covering different question types. This setting tests transfer across series, questions, and answer requirements without target-side training, tuning, or reference-answer feedback.

We compare TSHARNESS with three groups of baselines. • Task-specialized models include the released ChatTime (Wang et al., 2025a) and Time-MQA checkpoints (Kong et al., 2025). The zero-shot baseline directly answers each question from the input time series and accompanying context, without access to the harness tools or the memory. • To assess TSHARNESS against agentic approaches, we include chain-of-thought (CoT) (Wei et al., 2022), ReAct (Yao et al., 2023), and the released TimeClaw framework (Li et al., 2026). • We evaluate GPT-5.6-sol and the open-source Qwen3.8-27B. We implement CoT, ReAct, and TimeClaw, using Qwen3.8-27B as the backbone.

We report accuracy (%) across each benchmark, computed as the proportion of questions correctly answered under its native parsing and scoring protocols. Model responses that are invalid, unparseable, or format-violating are strictly penalized as incorrect. Appendix A.4 provides full details on answer parsing rules and metric accounting.

Table 1: **Cross-dataset zero-shot TSQA evaluation** (Accuracy, %). TSHARNESS is developed exclusively on the source benchmark (TSRBench) and deployed for zero-shot evaluation on five unseen target benchmarks. TSHARNESS* incorporates the learned RL-based TOOL SELECTOR. Shaded orange cells indicate **in-domain** training for ChatTime and Time-MQA, directly contrasting with the strictly zero-shot deployment of TSHarness. The best results within each backbone group are highlighted in **bold**; second-best results within the open-weight agentic group are underlined.

Method	Base LLM	TSRBench	Target Benchmarks				
			Time-MQA	ChatTime	CaTSBench	TSAQA	TimeOmni
Specialized Time-Series Models							
ChatTime	LLaMA-2-7B	19.95	28.79	88.77	9.63	11.36	19.66
Time-MQA	Qwen2.5-7B	30.47	54.98	37.07	26.05	45.07	32.04
	Llama3-8B	20.95	54.58	33.34	51.27	40.30	33.25
	Mistral-7B	36.27	55.91	35.47	47.79	42.61	31.55
Proprietary Model							
Zero-shot	GPT-5.6-sol	66.75	61.65	89.73	98.79	57.77	**75.73**
TSHarness	GPT-5.6-sol	**72.09**	**67.48**	**96.88**	**99.27**	**58.50**	74.76
Open-Weight Model and Agentic Methods							
Zero-shot	Qwen3.8-27B	43.69	58.01	74.13	92.96	52.18	50.97
CoT	Qwen3.8-27B	37.14	61.65	74.03	93.69	51.94	50.97
ReAct	Qwen3.8-27B	37.62	61.89	77.67	91.50	46.12	42.23
TimeClaw	Qwen3.8-27B	49.51	54.37	80.10	**96.84**	48.30	54.61
TSHarness	Qwen3.8-27B	**62.62**	**65.78**	96.10	96.60	59.47	**55.10**
TSHarness*	Qwen3.8-27B	**62.62**	**65.78**	**96.12**	**96.84**	**59.71**	**55.10**

4.2 MAIN RESULTS

TSHARNESS improves source-benchmark accuracy with both open-weight and proprietary backbones (Table 1). On TSRBench, accuracy increases from 43.69% to 62.62% with Qwen3.8-27B and from 66.75% to 72.09% with GPT-5.6-sol. The source-developed harness also transfers across datasets without target-side training or answer feedback. With Qwen3.8-27B, TSHARNESS improves over zero-shot answering on all five target benchmarks. The largest gain occurs on ChatTime, where accuracy rises from 74.13% to 96.10%. Gains on Time-MQA, CaTSBench, TSAQA, and TimeOmni are 7.77, 3.64, 7.28, and 4.13 percentage points, respectively. With GPT-5.6-sol, TSHARNESS improves accuracy on four of five targets, including increases from 61.65% to 67.48% on Time-MQA and from 89.73% to 96.88% on ChatTime. TimeOmni is the exception, with accuracy decreasing from 75.73% to 74.76%. Transfer benefits therefore extend across both backbones, although their magnitude and direction depend on the target benchmark.

Among methods using Qwen3.8-27B, TSHARNESS achieves the highest accuracy on TSRBench and four of the five target benchmarks. On TSRBench, it exceeds TimeClaw, the strongest baseline, by 13.11 percentage points. On Time-MQA, ChatTime, TSAQA, and TimeOmni, it exceeds the strongest baseline by 3.88, 16.00, 7.28, and 0.49 percentage points, respectively. On CaTSBench, TSHARNESS reaches 96.60%, compared with 96.84% for TimeClaw. These comparisons support the effectiveness of TSHARNESS across diverse TSQA tasks under a shared backbone. The Tool Selector variant, TSHARNESS*, is reported separately; Sec. 4.5 analyzes its execution cost and transfer results.

4.3 ABLATION STUDY OF TOOLS AND MEMORY

In Table 2, we isolate the individual contributions of numerical tools and procedural memory within TSHARNESS across both in-domain and zero-shot transfer settings. On the source benchmark (TSRBench), both components yield substantial performance gains. Disabling numerical tools precipitates a severe performance drop of 16.50 percentage points (62.62% $\rightarrow$ 46.12%), demonstrating that raw LLM reasoning alone cannot reliably parse intricate numerical patterns. Removing both components lowers accuracy to 42.48%. In cross-dataset zero-shot evaluation, Tool consistently improves accu-

Table 2: **Ablation study of core components across in-domain and zero-shot target datasets** (Accuracy, %). We evaluate variants of TSHarness by disabling numerical tools and procedural memory using Qwen3.8-27B as the backbone LLM. Numerical tool orchestration proves indispensable for both in-domain and transfer accuracy. Best results are shown in **bold**.

Variant	In-domain	Zero-shot				
	TSRBench	Time-MQA	ChatTime	CaTSBench	TSAQA	TimeOmni
Full TSHarness	**62.62**	65.78	96.10	**96.60**	**59.47**	**55.10**
w/o Tool	46.12	60.92	63.59	91.02	47.33	52.43
w/o Memory	60.44	**68.45**	**96.12**	96.12	58.74	54.37
w/o Tool + Memory	42.48	60.92	63.35	91.02	48.79	52.91

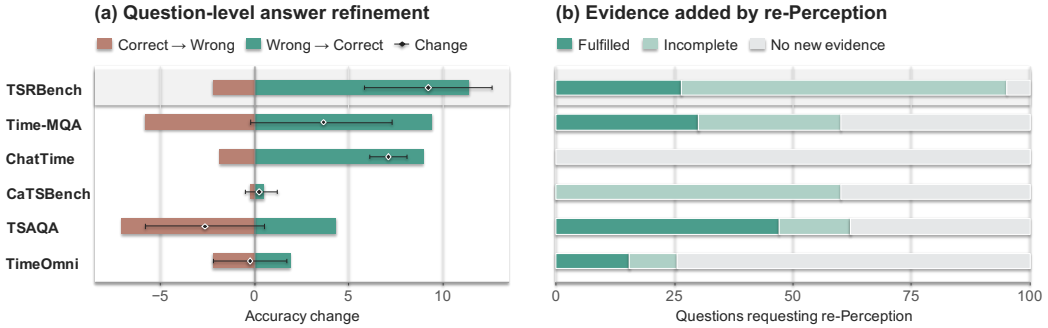

Figure 3: **Analysis of answer refinement and active re-perception dynamics.** (a) Transitions from initial pre-TPS hypotheses to final verified answers across benchmarks. Diamond markers represent net accuracy changes in percentage points. (b) Evidence fulfillment rates among queries triggering the re-perception loop (%). *Fulfilled* and *Incomplete* indicate acquired evidence with full and partial request coverage, respectively. Gray indicates that no additional evidence was obtained through re-perception.

racy across all five target benchmarks. Removing Tool lowers average target accuracy from 74.61% to 63.06%. Memory provides limited gains in this setting: average target accuracy is 74.76% without Memory, compared with 74.61% for the full harness. The full harness exceeds the variant without either component by 11.21 points in average target accuracy. These results establish Tool as a key contributor to cross-dataset transfer, while Memory is most beneficial within its construction domain.

4.4 Answer Refinement and Re-Perception

We analyze how the decoupled workflow refines initial reasoning and acquires incremental evidence, using GPT-5.6-sol rollouts from Table 1. Results are shown in Figure 3.

Question-level answer refinement. Figure 3(a) illustrates the transitions from ungrounded pre-TPS hypotheses to final grounded predictions. Across the majority of benchmarks, error corrections substantially surpass regressions, yielding significant net accuracy gains of $+9.22$ percentage points on TSRBench and $+7.10$ on ChatTime. Time-MQA shows the same positive trend, yielding a net accuracy gain of 3.64 points. While TSAQA and TimeOmni exhibit minor declines of -2.67 and -0.24 points, the overall results show that numerical perception and TPS-grounded reasoning improve initial judgments across several benchmarks, although the benefit varies across tasks.

Evidence added by re-perception. When the Answerer detects that existing state observations are insufficient to resolve ambiguous temporal queries, the harness activates targeted re-perception. As shown in Figure 3(b), the active feedback loop successfully acquires and fulfills missing evidence for 95.2% of requesting queries on TSRBench, 62.1% on TSAQA, and 25.5% on TimeOmni. Notably, ChatTime triggers zero re-perception requests; rather than a limitation, this indicates the cost-conscious selectivity of our harness, as the base perception state already provides sufficient quantitative support for its conversational queries. The harness loop thus functions as an adaptive, on-demand analytical mechanism that selectively expends computational budget only when resolving genuine evidentiary deficits. Detailed transition matrices and qualitative execution cases are provided in Appendix D.2 and Appendix B.5.

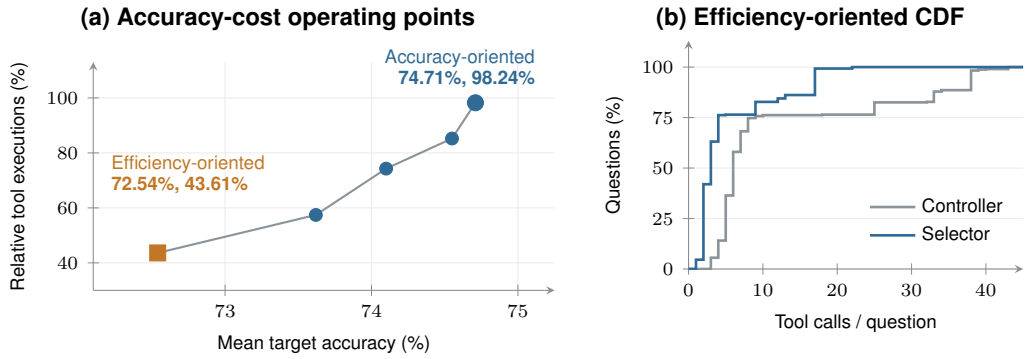

Figure 4: **Deployment choices for learned tool selection.** (a) Mean accuracy and total tool-execution ratios across all five target benchmarks (matched Controller = 100%). All five points are measured configurations; lines connect adjacent points. (b) The separate TSRBench cumulative distribution under efficiency-oriented deployment.

4.5 Analyses on Learned Tool Selection

The learned Tool Selector enables TSHarness to dynamically steer evidence acquisition across distinct accuracy and efficiency regimes at test time without retraining the policy in the target domain.

As shown in Figure 4(a), empirical evaluations across five operating configurations demonstrate this flexible control across all target benchmarks. Under the accuracy-oriented deployment (TSHarness*), the harness attains 74.71% macro-average target accuracy while reducing tool executions by 1.76% relative to the validated proposal baseline. Under the efficiency-oriented regime, the selector slashes overall tool executions by $\mathbf{56.39}\%$ while incurring only a modest 2.17% decrease in accuracy (72.54%). Throughout these comparisons, all tool costs are normalized to the matched Controller (the Perceiver's validated plan), with downstream evidence processing fixed within each comparison. These trajectories highlight that the Tool Selector provides effective, fine-grained budget control, illustrating that aggressive computational savings can be achieved with minimal degradation in reasoning fidelity.

Furthermore, distributional analysis on the source benchmark (Figure 4(b)) reveals that efficiency-oriented selection substantially suppresses the heavy upper tail of tool invocations. By penalizing redundant queries on complex, multi-series inputs, the learned policy effectively mitigates worst-case execution latency and bounds peak resource consumption. Together, learned preferences and evidence-preservation constraints equip TSHarness with configurable, cost-predictable perceptual efficiency across deployment environments. Details are documented in Appendices C.3 and D.4.

5 Conclusion and Limitations

We presented TSHarness, an agentic framework that makes source-developed analytical capabilities reusable across TSQA datasets. To overcome the dual transfer bottlenecks caused by coupled series and task shifts, our core design decouples continuous numerical perception from context-specific reasoning via an intermediate Time-Series Perception State (TPS). Within this framework, collaborative Perceiver and Answerer roles coordinate tool execution, procedural memory, and active evidence-request loops, while a reinforcement-learning Tool Selector adaptively optimizes the trade-off between answer accuracy and tool execution budget. Extensive evaluations demonstrate that an analytical workflow constructed exclusively on a single source dataset reliably generalizes across five unseen target benchmarks with diverse data distributions and answer specifications.

Despite its strong transferability, TSHarness has three main limitations. First, its perceptual capabilities are constrained by the pre-registered operations in its tool suite. Second, our Tool Selector uses a frozen policy network during zero-shot deployment, leaving test-time adaptation under severe domain shifts unexplored. Finally, zero-shot performance inherently depends on the diversity and task overlap of the source benchmark, highlighting the need for larger, open-world TSQA datasets to construct the harness system and evaluate broader generalizability.

AI USE STATEMENT

Generative AI tools and large language models were utilized exclusively for language polishing, grammatical refinement, and schematic figure drafting during manuscript preparation. AI was not employed to conceptualize research questions, design the core methodology, conduct physical or simulated experiments, or interpret scientific results. All algorithmic formulations, experimental analyses, and manuscript contents were conceived and verified by the authors, who assume full intellectual responsibility for this work. API-based LLM components of the evaluated system were accessed through official provider APIs, as described in Appendix A.

ETHICS STATEMENT

This work studies time-series question answering using existing benchmark datasets. Incorrect temporal interpretations or forecasts can mislead downstream decisions. TSHARNESS records tool evidence and provenance to support answer inspection; consequential applications require domain-specific validation and human oversight. Data use and redistribution should follow the original licenses and applicable privacy requirements.

REPRODUCIBILITY STATEMENT

Sec. 3 describes the runtime and Tool Selector. Sec. 4 and Appendix A specify the datasets, evaluated subsets, source/target separation, baselines, and inference configurations. Appendix A.4 defines the native answer metrics, and Appendix C records selector features, optimization settings, and cost accounting. Appendix B.1 describes the source construction of Tool and Memory resources, and Appendix D provides the execution breakdowns.

REFERENCES

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

Jiazhan Feng, Shijue Huang, Xingwei Qu, Ge Zhang, Yujia Qin, Baoquan Zhong, Chengquan Jiang, Jinxin Chi, and Wanjun Zhong. ReTool: Reinforcement learning for strategic tool use in LLMs. In *International Conference on Learning Representations*, pp. 37909–37926, 2026a.

Kun Feng, Ziwei Shan, Yuchen Fang, Yiyang Tan, Sihan Lu, Shuqi Gu, Xingyu Lu, Lintao Ma, and Kan Ren. KairosAgent: Agentic time series forecasting with fused semantic reasoning. *arXiv preprint arXiv:2605.30002*, 2026b. Accepted to EMNLP 2026.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. PAL: Program-aided language models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 10764–10799. PMLR, 2023.

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

Tong Guan, Zijie Meng, Dianqi Li, Shiyu Wang, Chao-Han Huck Yang, Qingsong Wen, Zuozhu Liu, Sabato Siniscalchi, Ming Jin, and Shirui Pan. TimeOmni-1: Incentivizing complex reasoning with time series in large language models. In *International Conference on Learning Representations*, pp. 152139–152170, 2026.

Malgorzata Gwiazda, Yifu Cai, Mononito Goswami, Arjun Choudhry, and Artur Dubrawski. TimeSeriesExamAgent: Creating time series reasoning benchmarks at scale. In *International Conference on Learning Representations*, pp. 86630–86671, 2026.

Liying Han, Kang Yang, Oliver Wang, Jason Wu, Pengrui Quan, Gaofeng Dong, Ozan Baris Mulayim, Sizhe Ma, Yuyang Yuan, Dezhi Hong, Mario Berges, and Mani Srivastava. TS-Skill: A benchmark for evaluating analytical skills in time-series question answering. *arXiv preprint arXiv:2605.24703*, 2026.

Ming Jin, Shiyu Wang, Lintao Ma, Zhixuan Chu, James Y. Zhang, Xiaoming Shi, Pin-Yu Chen, Yuxuan Liang, Yuan-Fang Li, Shirui Pan, and Qingsong Wen. Time-LLM: Time series forecasting by reprogramming large language models. In *International Conference on Learning Representations*, pp. 23857–23880, 2024.

Baoyu Jing, Sanhorn Chen, Lecheng Zheng, Boyu Liu, Zihao Li, Jiaru Zou, Tianxin Wei, Zhining Liu, Zhichen Zeng, Ruizhong Qiu, Xiao Lin, Yuchen Yan, Dongqi Fu, Jingchao Ni, Jingrui He, and Hanghang Tong. TSAQA: Time series analysis question and answering benchmark. In *Proceedings of the Fifth Workshop on Generation, Evaluation and Metrics (GEM)*, pp. 944–979. Association for Computational Linguistics, 2026.

Sehoon Kim, Suhong Moon, Ryan Tabrizi, Nicholas Lee, Michael W. Mahoney, Kurt Keutzer, and Amir Gholami. An LLM compiler for parallel function calling. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 24370–24391. PMLR, 2024.

Yaxuan Kong, Yiyuan Yang, Yoontae Hwang, Wenjie Du, Stefan Zohren, Zhangyang Wang, Ming Jin, and Qingsong Wen. Time-MQA: Time series multi-task question answering with context enhancement. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 29736–29753. Association for Computational Linguistics, 2025.

Zihao Li, Kaifeng Jin, Yuanchen Bei, Jiaru Zou, Avaneesh Kumar, Xuying Ning, Yanjun Zhao, Mengting Ai, Baoyu Jing, Hanghang Tong, and Jingrui He. Harnessing generalist agents for contextualized time series. *arXiv preprint arXiv:2606.05404*, 2026.

Penghang Liu, Elizabeth Fons, Annita Vapsi, Mohsen Ghassemi, Svitlana Vyetrenko, Daniel Borrajo, Vamsi K. Potluru, and Manuela Veloso. TS-Agent: Understanding and reasoning over raw time series via iterative insight gathering. *arXiv preprint arXiv:2510.07432v2*, 2026.

Xu Liu, Junfeng Hu, Yuan Li, Shizhe Diao, Yuxuan Liang, Bryan Hooi, and Roger Zimmermann. UniTime: A language-empowered unified model for cross-domain time series forecasting. In *Proceedings of the ACM Web Conference 2024*, pp. 4095–4106. ACM, 2024.

Cheng Qian, Emre Can Acikgoz, Qi He, Hongru Wang, Xiusi Chen, Dilek Hakkani-Tür, Gokhan Tur, and Heng Ji. ToolRL: Reward is all tool learning needs. In *Advances in Neural Information Processing Systems*, volume 38, pp. 105523–105553, 2025.

Benjamin Reichman, Xiaofan Yu, Lanxiang Hu, Jack Truxal, Atishay Jain, Rushil Chandrupatla, Tajana Šimunić Rosing, and Larry Heck. SensorQA: A question answering benchmark for daily-life monitoring. In *Proceedings of the 23rd ACM Conference on Embedded Networked Sensor Systems*, pp. 282–289. ACM, 2025.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems*, volume 36, pp. 68539–68551, 2023.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Yu Shang, Yu Li, Keyu Zhao, Likai Ma, Jiahe Liu, Fengli Xu, and Yong Li. AgentSquare: Automatic LLM agent search in modular design space. In *International Conference on Learning Representations*, pp. 3841–3865, 2025.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, pp. 8634–8652, 2023.

Chengsen Wang, Qi Qi, Jingyu Wang, Haifeng Sun, Zirui Zhuang, Jinming Wu, Lei Zhang, and Jianxin Liao. ChatTime: A unified multimodal time series foundation model bridging numerical and textual data. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(12):12694–12702, 2025a.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *Transactions on Machine Learning Research*, 2024.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 63897–63911. PMLR, 2025b.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pp. 24824–24837, 2022.

Gerald Woo, Chenghao Liu, Akshat Kumar, Caiming Xiong, Silvio Savarese, and Doyen Sahoo. Unified training of universal time series forecasting transformers. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 53140–53164. PMLR, 2024.

Zhe Xie, Zeyan Li, Xiao He, Longlong Xu, Xidao Wen, Tieying Zhang, Jianjun Chen, Rui Shi, and Dan Pei. ChatTS: Aligning time series with LLMs via synthetic data for enhanced understanding and reasoning. *Proceedings of the VLDB Endowment*, 18(8):2385–2398, 2025.

Binfeng Xu, Zhiyuan Peng, Bowen Lei, Subhabrata Mukherjee, Yuchen Liu, and Dongkuan Xu. ReWOO: Decoupling reasoning from observations for efficient augmented language models. *arXiv preprint arXiv:2305.18323*, 2023.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-Mem: Agentic memory for LLM agents. In *Advances in Neural Information Processing Systems*, volume 38, pp. 17577–17604, 2025.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023.

Yao Yin, Zhenyu Xiao, Musheng Li, Yiwen Liu, Sutong Nan, Yiting He, Ruiqi Wang, Zhenwei Zhang, Qingmin Liao, and Yuantao Gu. MMTS-BENCH: A comprehensive benchmark for time series understanding and reasoning. *arXiv preprint arXiv:2602.08588*, 2026.

Fangxu Yu, Xingang Guo, Lingzhi Yuan, Haoqiang Kang, Hongyu Zhao, Lianhui Qin, Furong Huang, Bin Hu, and Tianyi Zhou. TSRBench: A comprehensive multi-task multi-modal time series reasoning benchmark for generalist models. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026.

Tianxiang Zhan, Xiaobao Song, Tong Guan, Shirui Pan, and Ming Jin. AION: Next-generation tasks and practical harness for time series. *arXiv preprint arXiv:2605.25045*, 2026.

Luca Zhou, Pratham Yashwante, Marshall Fisher, Alessio Sampieri, Zihao Zhou, Fabio Galasso, and Rose Yu. CaTS-Bench: Can language models describe time series? In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 34479–34519. Association for Computational Linguistics, 2026.

A Datasets and Evaluation Details

This section supplements the experimental setup in Sec. 4, specifying the QA tasks, input representation, evaluation subsets, and answer scoring.

A.1 Benchmark coverage

Table 3 summarizes the evaluated QA tasks. TSRBench (Yu et al., 2026) supplies the construction data and in-domain evaluation. ChatTime (Wang et al., 2025a), Time-MQA (Kong et al., 2025), CaTSBench (Zhou et al., 2026), TSAQA (Jing et al., 2026), and TimeOmni-1 (Guan et al., 2026) test transfer across series, contexts, and answer requirements.

Table 3: Time-series inputs, QA capabilities, and answer forms covered by the evaluation.

Benchmark	Time series and context	QA capabilities	Answer forms
TSRBench	Single or multiple series with domain context	Perception, reasoning, prediction, and decision-making	Multiple choice
ChatTime	Synthetic series of varying lengths	Trend, volatility, seasonality, and outlier recognition	Multiple choice
Time-MQA	Real-world series with contextual descriptions	Statistical properties, patterns, anomalies, and structural changes	True/false; multiple choice
CaTSBench	Series paired with textual descriptions	Caption and series retrieval, description consistency, and pairwise comparison	True/false; multiple choice
TSAQA	Cross-domain series and temporal patches	Classification, anomalies, characterization, comparison, transformations, and temporal relationships	True/false; multiple choice; ordered labels
TimeOmni-1	Series with event and domain context	Causality discovery, scenario understanding, and decision-making	Multiple choice

A.2 Input representation and transfer protocol

We use the input notation $u = (X, c, \kappa)$ introduced in Sec. 3:

- **Observations X.** Dataset adapters separate numerical observations from the question, retaining the correspondence between each series and its description. Tools operate on these observations; the Controller uses the bounded view described in Sec. 3.
- **Context c.** The question, domain descriptions, and answer choices remain available for planning and contextual reasoning. Retrieval and comparison questions retain their candidate series or descriptions; temporal-ordering questions retain their patch identities.
- **Requirements κ.** The requested answer form determines output parsing. Reference answers provide source feedback and determine evaluation scores.

TSRBench construction examples provide feedback for developing the tools and organizing analysis knowledge into Memory. In-domain evaluation uses the held-out approximately 10% TSRBench subset following the TimeClaw protocol (Li et al., 2026). For each backbone, the entire system is fixed before target evaluation, including model weights, all prompts, tools and activation rules, Memory and retrieval rules, the Tool Selector when used, dataset adapters, evidence-requirement mappings, and answer parsers. The analysis remains adaptive at inference through tool plans and evidence requests, as specified in Eq. (1). The independent zero-shot baseline receives the series and question directly, without Tool or Memory access.

Source-data roles. Within TSRBench, construction examples supply execution traces and answer feedback for developing Tool and Memory resources. The Tool Selector uses a source-training subset for policy updates and a separate source-development subset for checkpoint selection. Its configuration is given in Appendix C; the resource-construction procedure is detailed in Appendix B.1. The held-out TSRBench test partition is excluded from resource construction, policy training, and checkpoint selection. Target benchmarks are excluded from system development and tuning; their reference answers are used only to score completed predictions. This separation prevents evaluation data from entering system development. The TSRBench test partition measures ID performance, while the other five benchmarks assess transfer of the source-developed system.

Inference configuration. API-based LLM components use official provider APIs. The TSRBench harness runs use temperature 0 and a maximum of 768 generated tokens per LLM call for both backbones. Planning, TPS synthesis, and answering use separate role-specific calls to the same backbone. Structured responses receive one additional parsing attempt when needed. Tool-plan and evidence-request budgets are specified in Appendix B.4.

A.3 Evaluation subsets

For cross-evaluation, we use the full test sets provided by prior work. For comparisons among agentic methods, we randomly sample a fixed subset of 412 questions per benchmark, covering different question types, and evaluate the compared methods on the same subset. The sampled subsets for agentic comparisons on the target benchmarks are described below.

- **ChatTime.** We randomly sample 412 questions covering four task categories: Trend, Seasonality, Volatility, and Outliers.
- **Time-MQA.** We randomly sample 412 questions from the objective test set, covering different question types.
- **CaTSBench and TSAQA.** Each benchmark uses a fixed subset of 412 randomly sampled questions covering different task categories. CaTSBench plot-retrieval questions requiring unavailable candidate images are excluded from the text-based evaluation; numerical-series and text-description tasks are retained.
- **TimeOmni-1.** We randomly sample 412 accuracy-scored questions from the OOD split, covering causality discovery, scenario understanding, and decision-making. All of these questions are target data relative to TSRBench construction.

Comparisons among agentic methods on TSRBench use the same held-out 412-question test subset.

A.4 Answer parsing and scoring

Let P_d denote the answer parser for benchmark d, $\hat{y}_i$ the generated response, and y_i the reference. Accuracy is

$$\mathrm{Acc}_d = \frac{100}{N_d} \sum_{i=1}^{N_d} \mathbf{1}[P_d(\hat{y}_i) = P_d(y_i)] \,, \tag{8}$$

where N_d is the number of completed inference runs included in scoring. An invalid or unparseable final response from a completed run contributes zero.

- **Multiple choice.** Responses are mapped to an option label using the benchmark's answer format and compared with the reference label.
- **True/false.** Boolean answers are normalized before exact comparison.
- **Temporal ordering.** Exact agreement of the complete ordered answer with the reference is required after text normalization.

When reporting an average over target benchmarks, we use the unweighted mean $\overline{\mathrm{Acc}} = \frac{1}{5}\sum_{d\in\mathcal{D}_{\mathrm{target}}} \mathrm{Acc}_d$, excluding TSRBench. Percentage-point differences are computed from unrounded accuracies.

B HARNESS IMPLEMENTATION DETAILS

This section supplements Section 3 with the tool interfaces, construction of the intermediate TPS, and evidence updates underlying the decoupling of perception and answering. Two successful target-benchmark executions illustrate how numerical findings support contextual judgments.

B.1 SOURCE CONSTRUCTION AND FROZEN DEPLOYMENT

The development stage in Figure 2 converts TSRBench execution feedback into the Tool and Memory resources used by the harness. Its data roles are specified in Appendix A. Construction proceeds in batches of 16 source questions:

1. **Run and inspect.** The current harness processes each question and records its tool evidence, TPS, preliminary hypothesis, and final prediction. Reference answers are then used to assess the completed batch and distinguish corrected, regressed, and persistently incorrect judgments.
2. **Develop the tool configuration.** The initial library combines general statistical and temporal operations with implemented task-specific calculations, such as signal analysis, formula evaluation, and candidate-trajectory comparison. Source development includes implementing new operations, configuring existing tools, and establishing tool–task activation rules. Source feedback determines the available specialized tool–task pairings and motivates additional diagnostic operations. For calculation-based pairings, the activation rule requires at least two supporting observations and 95% agreement with source references. Diagnostic operations that return measurements or interpretations are enabled through checks on source examples; their outputs enter TPS for the Answerer to interpret.
3. **Organize analytical knowledge.** An LLM editor turns the batch's evidence and outcomes into Memory entries containing a general procedure, applicability conditions, recommended tools, and evidence requirements. Entries are organized by task or analytical topic. An update revises the corresponding entry when that topic already exists. Checks validate the topic and tool references and filter literal option-answer statements; task-specific checks assess proposed rules against the available source evidence.
4. **Freeze the resources.** The resulting Memory bank and enabled-tool configuration are saved for deployment. The Tool Selector is subsequently trained with these resources fixed, as detailed in Appendix C. Target inference loads the source-developed resources and computes measurements for each new question through the same tool interfaces and TPS schema.

B.2 TOOLS AND ANALYTICAL GUIDANCE

Tool interfaces and procedural guidance. The library covers statistics, trends, seasonality, volatility, change points, and cross-series relationships. Eligibility checks use declared requirements, including minimum sequence length and the availability of multiple channels. Plans specify tool names and supported arguments, which can identify a channel or temporal interval. The runtime checks names, argument types, and declared ranges before execution. Memory entries describe applicability, recommended operations, and evidence requirements. Their summaries inform planning, and a query generated with the plan retrieves relevant procedures under fixed retrieval rules.

Table 4 summarizes representative numerical operations and their outputs. Each registered tool also declares supported fields, an argument schema, and applicability conditions. For example, trend estimation requires at least two observations, while cross-series analysis requires multiple channels and at least four observations. Validated arguments can adjust detection thresholds or the maximum lag. The Controller's plan is deduplicated and restricted to eligible operations before execution.

Memory contents. An entry stores analysis guidance together with its applicability, triggers, recommended tools, and required evidence. Retrieval matches the query against the entry's content, tags, and triggers. For example, one source-developed entry explains that comparing the innovation scales of two AR(1) series requires fitting each autoregression and comparing residual standard deviations. It recommends the corresponding signal-analysis tool and identifies residual scale and

Table 4: Representative tools used to construct TPS fields. Measurements and temporal locations are computed from the input observations.

Tool family	Computation	Returned information
Statistics and value profile	Summary statistics, quantiles, extrema, and adjacent differences	Values, extrema locations, largest rise and drop
Trend	Linear fit on normalized time; slope scaled by series standard deviation	Direction, strength, normalized slope
Seasonality	Autocorrelation and spectral analysis	Period and periodicity strength
Volatility	Early/late standard deviations and their ratio	Variability level and change
Anomalies and change points	Median-deviation and local-residual scores; mean and variance changes across windows	Event positions, types, directions, and scores
Cross-series	Contemporaneous and lagged correlations	Channel pairs, correlations, and strongest lag

autoregressive coefficient as the required evidence. This makes the relevant analytical steps available to the Controller while the tools compute their values for the current input.

B.3 From tool outputs to TPS

Table 5 expands the statistical fields, temporal fields, and named extensions defined in Eq. (3). Field values may be scalars or structured outputs; supporting evidence identifiers link them to the producing tool records. Series identity, channels, and length remain available alongside the fields, and the LLM-generated summary provides a concise textual view.

Table 5: TPS field groups and their analytical roles.

Component	Contents and role
Statistics $\mathcal{S}_j$	Distributions and value profiles, including means, standard deviations, extrema, and their locations.
Temporal features $\mathcal{F}_j$	Trends, periodicity, volatility, segments, changes, anomalies, and cross-series relationships.
Task extensions $\mathcal{D}_r$	Named analyses that add evidence required by the question beyond the common fields.

Building the intermediate representation. Perception uses two role-specific LLM calls: the first proposes tools and a Memory query; the second receives the resulting evidence, selects measured fields, and writes the summary h_j. TPS construction then proceeds as follows:

- **Bind values to measurements.** Each selected field is populated directly from its tool record. A value can be a scalar, a structured property such as trend direction and strength, or a list of events with their temporal positions.
- **Retain analytical meaning and sources.** Each field stores its supporting evidence identifiers, the producing tool and version, and available confidence information. The series identity and original indices connect temporal findings back to the observations.
- **Represent task-specific analysis.** Named extensions store the requested operations, returned evidence, and coverage status in the form $\mathcal{D}_r[\nu] = (q, V, I, \delta)$ from Section 3.3. These can include cross-series comparisons, candidate trajectory measurements, or temporal-ordering scores.

Thus, the Answerer receives an interpretable description of what was measured and where, together with question-specific evidence. It uses the question and answer requirements to turn this description into a contextual judgment.

Initial hypothesis and task extensions. Before evidence-conditioned answering, a separate call to the same backbone receives the raw numerical series, question context, answer choices, and output requirements, without TPS evidence or reference answers. Its parsed candidate answer and response are stored in a named extension as a provisional hypothesis, separate from the measured core properties. The Answerer reassesses this hypothesis against the acquired evidence. An initial extension-planning call uses the core findings, question, evidence requirements, and available tools and Memory to propose additional analyses; the harness validates and executes these requests to populate $\mathcal{D}_0$.

Role-specific LLM interfaces. Table 6 summarizes the information passed between the LLM roles using the same backbone. Memory summaries inform the tool plan, and retrieved entries guide the interpretation of measured evidence. The LLM selects fields and writes the summary; the harness populates their numerical values directly from tool records. The Answerer then interprets TPS under the question's context and output requirements.

Table 6: Inputs and outputs of the LLM roles connecting Perception, TPS, and the Answerer. Tool execution and numerical field assignment are performed by the harness between calls.

LLM role	Inputs	Outputs
Perception planning	Bounded series view, question context, eligible tools, Memory summaries, and execution limits.	Tool names and arguments; a Memory query and retrieval choices.
TPS synthesis	Tool evidence, retrieved Memory entries, series identity, channels, length, and question context.	Measured fields to include and a factual summary h_j describing the series.
Initial hypothesis	Raw numerical series, question context, answer choices, and output requirements.	A parsed candidate answer and its generated response, stored as an initial hypothesis in a named extension.
Initial extension planning	Core findings, question, evidence requirements, and available tools and Memory.	Analysis requests specifying operations, arguments, and required evidence.
Answerer	Question and answer requirements, TPS with the provisional hypothesis, evidence coverage, available operations, and request budget.	A task-formatted prediction with evidence references, or a request specifying the additional evidence needed.

B.4 Evidence updates in the harness loop

The implementation maintains the core fields and accumulates named extensions through the update in Eq. (5).

- **Specify the need.** A request identifies the series, operations, arguments, and evidence types needed for further reasoning.
- **Execute and attach.** The harness validates the request against available operations and the request budget, executes the analysis, and attaches the results with their sources to a new extension.
- **Reassess the state.** Coverage is computed from available fields and returned evidence types. The Answerer uses the updated TPS to continue reasoning or produce the task-formatted output. The extension's coverage status records whether the requested evidence types were obtained.

Runtime limits. The source configuration gives the Perception Controller a view of at most 48 observations per series, while numerical tools use the original sequence. Base perception admits up to nine tools per series; the initial planner adds at most one task extension per question. The Answerer can make one accepted evidence request within a three-step decision budget. Requested raw-series views are bounded to 128 points across the supplied channels. These limits are shared by the two conditions in the matched Tool Selector comparison.

B.5 Successful executions on target benchmarks

The following examples are drawn from completed Qwen3.8-27B trajectories using the TSRBench-developed harness. Both final predictions match the dataset references.

ChatTime: identifying a sustained level shift. The question asks whether a 128-point series has no obvious outlier, a sudden spike, or a level shift. The relevant evidence is compact:

- **Value profile:** the largest adjacent drop is -2.48, from index 31 to 32 (zero-based indexing).
- **Temporal features:** the change-point tool detects a downward level shift at index 32; the trend tool reports a strong downward trend.
- **Task extension:** an outlier-type analysis collects anomaly and change-point evidence for interpreting the observed change.

The Answerer combines the abrupt drop with the persistent change in level and selects *C: level shift*. This execution illustrates the separation between locating and measuring a change in Perception and interpreting its type under the question's definitions.

TSAQA: reconstructing a temporal sequence. An electricity-demand question provides the first patch and four remaining patches, each containing 45 observations, and asks for their chronological order. A TPS extension evaluates how adjacent patches join. For patches u and v, its boundary cost is

$$c(u, v) = \frac{|u_{\text{last}} - v_{\text{first}}| + 0.25\,|\Delta u_{\text{end}} - \Delta v_{\text{start}}|}{s + \epsilon}, \tag{9}$$

where s is the standard deviation across all supplied patches, the two differences are the last and first adjacent increments, and $\epsilon = 10^{-12}$. Candidate orders are ranked by the sum of boundary costs, including the join from the given first patch. The extension records a cost of 0.642 for *B,C,A,D*, compared with 1.916 for the next candidate. Using this evidence, the Answerer revises its preliminary hypothesis *D,A,B,C* to the correct sequence *B,C,A,D*. The same TPS interface therefore supports both a categorical judgment in ChatTime and an ordered output in TSAQA.

C Tool Selector Training Details

This section specifies the feature representation, optimization, and deployment of the Tool Selector in Sec. 3.5. The configuration below is used in the matched Qwen3.8-27B base-tool comparison. The selector refines each series' initial tool plan; the resulting evidence is processed by the same TPS construction and answering procedures described in Appendix B.

C.1 Policy inputs and network

For candidate tool i at decision t, the implementation constructs

$$\phi_{t,i} = [h(c_t); h(d_i); h(c_t) \odot h(d_i); v_{t,i}] \in \mathbb{R}^{396}, \tag{10}$$

where c_t combines the question with its task and dimension descriptors, d_i contains the tool name, description, supported operations, and output fields, and $\odot$ is elementwise multiplication. The function h maps lowercase text tokens into a fixed 128-dimensional signed hash vector and normalizes its length. The three text blocks therefore contribute 384 dimensions. The remaining 12 scalar features comprise:

- **Plan and tool information:** membership in the Perceiver proposal, proposal and candidate-set sizes, declared tool cost and reliability, multivariate applicability, text-vector similarity, and a constant feature.
- **Current-series information:** scaled sequence length and channel count, standardized endpoint change, and first-difference variability relative to the series standard deviation. The latter two are bounded with $\tanh$.

These features provide a compact observation of the decision state in Eq. (6). A shared multilayer perceptron, $396 \to 64 \to 1$ with a $\tanh$ hidden activation, outputs the adjustment $f_\theta(\phi_{t,i})$ for each candidate.

C.2 Perceiver prior and subset sampling

The initial inclusion probability is 0.8 for a tool proposed by the Perceiver and 0.2 for another eligible tool. Zero initialization of the final linear layer makes the initial policy follow these priors. The learned adjustment is added to the prior logit, and the resulting logit is bounded to $[-8, 8]$ before applying the sigmoid.

For the inclusion probabilities $p_{t,i}$ defined in Sec. 3.5, conditioning on a nonempty subset gives

$$\pi_\theta(a_t \mid s_t) = \frac{\prod_{i \in \mathcal{T}_t} p_{t,i}^{\mathbf{1}[i \in a_t]} (1 - p_{t,i})^{\mathbf{1}[i \notin a_t]}}{1 - \prod_{i \in \mathcal{T}_t} (1 - p_{t,i})}, \qquad a_t \neq \varnothing. \tag{11}$$

The denominator removes the empty subset and normalizes the remaining probabilities.

Training samples tool memberships from this joint policy, resampling an empty subset. The eligible pool fits within the nine-tool base-perception budget, so sampled subsets retain their full membership. In efficiency-oriented deployment, tools with inclusion probability at least 0.5 are executed; if none passes, the highest-probability tool is selected. Retained tools preserve their Perceiver-provided arguments and order, while additional tools use their registered defaults.

C.3 Deployment configurations

The deployment configurations use the deterministic thresholding policy from Sec. 3.5 with the same source-selected checkpoint. The efficiency-oriented configuration applies the inclusion rule directly. Accuracy-oriented deployment moderates the learned adjustment and applies evidence-preservation constraints after thresholding. Its inclusion probabilities retain the Perceiver prior:

$$p_{t,i}^{(\alpha)} = \sigma\big(\mathrm{clip}\big[\mathrm{logit}(p_{t,i}^0) + \alpha f_\theta(\phi_{t,i}), -8, 8\big]\big), \qquad \alpha = 0.4. \tag{12}$$

Here $p_{t,i}^0$ is the prior inclusion probability. Scaling the learned adjustment preserves the policy's tool-dependent preferences while reducing departures from the initial plan; α is not a fraction of tools to execute. No policy weights are updated during this adjustment.

After thresholding, the accuracy-oriented rule retains at least two eligible Perceiver-proposed tools, or all of them if fewer than two were proposed. A proposed `value_profile` is retained; when both `volatility` and `statistics` are proposed, both are retained. These constraints preserve basic numerical evidence while leaving the learned selector to filter the remaining operations. The deployment rules are fixed before target evaluation and apply uniformly across target questions, without target-side tuning or reference-answer feedback.

In the accuracy-oriented matched comparison, both conditions explicitly present task-relevant TPS measurements and instruct the Answerer to check its initial judgment against them. Evidence matching recognizes equivalent field names, and interval slicing preserves both observations of a two-point interval. Holding these procedures fixed evaluates the learned selector together with its conservative deployment constraints against the no-selector Controller. Across deployment configurations, inference rules and evidence processing can differ; their comparison characterizes deployment choices rather than isolating the effect of a single selector parameter.

C.4 ROLLOUT REWARD AND OPTIMIZATION

Each training question generates $G = 4$ complete harness rollouts. A rollout can contain several selection decisions when the question includes multiple series. For rollout g, the terminal reward is the correctness–cost objective in Eq. (7), with $\lambda_{\text{tool}} = 0.01$ and N_{base} summed across its series. Its advantage is

$$\widehat{A}_g = R_g - \frac{1}{G-1} \sum_{g' \neq g} R_{g'}, \tag{13}$$

using the other rollouts of the same question as the baseline. Advantages retain the original reward scale, preserving the relative weight of correctness and tool cost.

Let T_g be the number of selection decisions in rollout g. The likelihood ratio includes every selected subset in that rollout:

$$\rho_g(\theta) = \prod_{t=1}^{T_g} \frac{\pi_\theta(a_{g,t} \mid s_{g,t})}{\pi_{\text{old}}(a_{g,t} \mid s_{g,t})}. \tag{14}$$

Each subset probability includes the nonempty-set normalization in Eq. (11). The implementation minimizes

$$\mathcal{L}(\theta) = \mathbb{E}_g \Big[-\min\Big(\rho_g \widehat{A}_g, \text{clip}(\rho_g, 1-\epsilon, 1+\epsilon) \widehat{A}_g \Big) + \beta \overline{D}_g - \eta \overline{H}_g \Big], \tag{15}$$

where $\overline{D}_g$ averages the KL divergence between each candidate's Bernoulli inclusion probability and its Perceiver prior, and $\overline{H}_g$ averages the corresponding Bernoulli entropies. These regularizers operate on the individual inclusion distributions; the policy-gradient term uses the conditioned subset probabilities. Table 7 gives the training settings.

Table 7: Configuration of the learned Tool Selector in the matched base-tool experiment. The LLM and source-developed Tool and Memory resources remain fixed during policy optimization.

Setting	Value
Source training / development questions	96 / 48
Training passes	1
Rollouts per question / questions per batch	4 / 4
Text-hash dimension / scalar features	128 per block / 12
Policy network	$396 \to 64 \to 1$, tanh
Perceiver prior: proposed / other tools	0.8 / 0.2
Optimizer / learning rate	Adam / 3×10^{-3}
Update epochs per rollout batch	4
Policy-ratio clip ϵ	0.2
Prior KL weight β	0.02
Entropy weight η	0.001
Gradient norm cap	1.0
Cost weight λ_{tool}	0.01 per base-tool execution
Deployment inclusion threshold	0.5
Random seed	20260909

C.5 CHECKPOINT SELECTION AND MATCHED EVALUATION

We evaluate the predetermined halfway and final checkpoints on the source development questions. Checkpoints are ranked first by answer accuracy, then by mean correctness–cost reward, with an earlier checkpoint preferred if both tie. The final checkpoint is selected in the reported experiment. Deployment uses its deterministic inclusion rule.

The Controller condition executes the PERCEIVER's validated proposal without a selector; the RL condition applies the learned policy to refine that proposal. Both conditions use the same eligible tools, frozen backbone, Memory, extension rules, and evidence-request budget. Identical model requests share cached responses within the comparison, fixing model outputs for repeated prompts while allowing changed evidence to produce different requests. In this efficiency-oriented TSRBench comparison, Controller and selector accuracies are 61.89% and 59.95%, respectively. The 62.62%

harness result in Table 1 comes from the separate main evaluation. Figure 4(b) retains the execution distribution from this matched comparison.

The efficiency-oriented cross-dataset evaluation freezes the selected checkpoint and TSRBench-developed Tool and Memory resources and deploys them on 412 questions from each target benchmark. Target inference uses the same Qwen3.8-27B backbone with temperature 0, an 8,192-token context limit, and at most 768 generated tokens per model call. Target accuracy is evaluated with response caching disabled, using the completed-run scoring rule in Appendix A.4. Figure 4(a) aggregates all five target benchmarks; the TSRBench comparison above provides the separate per-question distribution in panel (b).

The accuracy-oriented comparison covers the same five target benchmarks: Time-MQA, ChatTime, CaTSBench, TSAQA, and TimeOmni. Its no-selector control uses the same evidence-processing procedures specified in Appendix C.3. Appendix D.4 reports the endpoint and intermediate results, with costs normalized to the corresponding matched Controller conditions.

Execution accounting. Following Eq. (7), the reward penalizes base-tool executions to reduce numerical analysis overhead while retaining answer correctness. Evaluation records all executed tools, summed over every series in a question:

$$N_{\text{all}} = N_{\text{base}} + N_{\text{question}} + N_{\text{planned}} + N_{\text{requested}}. \tag{16}$$

The additional terms count question-specific analytical tools, initial TPS-extension tools, and tools triggered by accepted Answerer requests. An evidence revisit counts an accepted request, which can execute multiple operations. Initial-hypothesis generation is counted as a model call. Logical model-call counts include cached responses and are tracked separately from tool executions; a tool that invokes the LLM contributes to both categories. Appendix D provides the corresponding empirical breakdowns.

D Supplementary Experimental Results

We supplement Sec. 4 with results across task categories, answer revisions after evidence integration, and the distribution of tool executions. Task-level scores use the same completed predictions as Table 1; the Tool Selector analysis uses the separate matched comparison specified in Appendix C.

D.1 Performance across capabilities and task types

Tables 8 and 9 compare the complete harness with direct zero-shot answering under each backbone. Benchmark-native categories preserve differences in analytical demands. Time-MQA is grouped by answer format because its detailed task labels are heterogeneous. Its evaluation subset follows the sampling procedure in Appendix A.

Source capabilities. On TSRBench, the Qwen-based harness improves all four capability groups. The largest increases occur in reasoning (36.26% to 65.50%) and decision-making (25.40% to 50.79%). The GPT-based harness also improves reasoning, perception, and decision-making, while prediction decreases from 61.11% to 57.41%. These results locate the aggregate source gain across distinct analytical demands.

Transfer across question types. The target results reveal gains in both temporal perception and task-specific interpretation:

- **Temporal properties.** On ChatTime, Qwen improves seasonality recognition from 60.70% to 91.20% and volatility recognition from 60.40% to 97.00%. GPT improves seasonality from 66.70% to 91.10%; both harnesses reach 100% on trend questions.
- **Relational and structured judgments.** On TSAQA, Qwen improves temporal relationships from 13.24% to 45.59% and data transformation from 39.71% to 55.88%. On CaTSBench, its peak-timing comparison rises from 70.59% to 91.18%, and volatility comparison from 76.47% to 88.24%.

Table 8: Accuracy (%) across source capabilities and target question categories, using the matched same-backbone predictions underlying Table 1. TSRBench is grouped by capability, ChatTime and TimeOmni-1 by task, and Time-MQA by answer format. Δ is TSHarness minus zero-shot in percentage points; bold indicates the higher score within each pair.

	Qwen3.8-27B			GPT-5.6-sol		
Capability / task	Zero-shot	TSHarness	Δ	Zero-shot	TSHarness	Δ
TSRBench						
Decision-making	25.40	**50.79**	+25.40	44.44	**53.97**	+9.52
Perception	68.57	**71.43**	+2.86	82.86	**87.14**	+4.29
Prediction	50.00	**59.26**	+9.26	**61.11**	57.41	-3.70
Reasoning	36.26	**65.50**	+29.24	71.93	**81.87**	+9.94
ChatTime						
Outliers	92.10	**96.20**	+4.10	96.20	**98.90**	+2.70
Volatility	60.40	**97.00**	+36.60	**97.60**	97.50	-0.10
Trend	83.30	**100.00**	+16.70	98.40	**100.00**	+1.60
Seasonality	60.70	**91.20**	+30.50	66.70	**91.10**	+24.40
Time-MQA						
Multiple choice	62.10	**64.92**	+2.82	59.68	**64.11**	+4.44
True/false	51.83	**67.07**	+15.24	64.63	**72.56**	+7.93
TimeOmni-1						
Causality discovery	40.34	40.34	0.00	51.70	**52.84**	+1.14
Scenario understanding	74.72	**83.71**	+8.99	**97.19**	94.94	-2.25
Decision-making	10.34	**12.07**	+1.72	**82.76**	79.31	-3.45

- **Contextual interpretation.** On TimeOmni-1, Qwen improves scenario understanding from 74.72% to 83.71%, accounting for most of its aggregate gain. GPT starts at 97.19% on this category and reaches 94.94% with the harness, illustrating the dependence of transfer gains on the backbone's initial capability.

The breakdowns connect the transfer results to the statistical, temporal, and task-specific information represented in TPS. Improvements span several question types, while categories such as TSAQA comparison and perturbed-series retrieval show that the benefit is not uniform.

D.2 Answer revisions after evidence integration

This section supplements the pipeline analysis in Sec. 4.4 with initial and final accuracies and evidence-request frequencies. The provisional hypothesis described in Sec. 3 provides a reference for examining how judgments change after TPS evidence is incorporated. For each completed trajectory, we score this hypothesis and the final prediction with the same benchmark parser. Let $\mathrm{Acc}_{\mathrm{initial}}$ and $\mathrm{Acc}_{\mathrm{final}}$ denote their accuracies in percent. The net change in percentage points is

$$\Delta_{\mathrm{revision}} = \mathrm{Acc}_{\mathrm{final}} - \mathrm{Acc}_{\mathrm{initial}}. \tag{17}$$

Table 10 reports initial and final accuracies and their net change for each benchmark and backbone. Figure 3(a) visualizes answer transitions and the net accuracy change for the GPT backbone. The 95% intervals for the net change are the 2.5th and 97.5th percentiles of 20,000 question-level bootstrap resamples within each benchmark, preserving the initial–final prediction pairs. These intervals describe question-level variation within the evaluated subsets.

The revisit column additionally records use of the evidence-request loop. GPT requests additional evidence on 19.90% of TSRBench questions and 48.54% of TimeOmni-1 questions, whereas Qwen mainly completes its judgments from the initial TPS. The same interface thus supports both direct reasoning over the initial state and further perception when requested.

With Qwen, final accuracy exceeds initial accuracy on every evaluated benchmark. On TSRBench, accuracy increases from 44.17% to 62.62%; on ChatTime, it increases from 74.30% to 96.10%; and on TSAQA, it increases from 52.18% to 59.47%. GPT also obtains net gains on TSRBench,

Table 9: Task-level accuracy (%) on CaTSBench and TSAQA. All evaluated task categories are included. Both conditions use the same questions within each benchmark. Bold and Δ follow Table 8.

	Qwen3.8-27B			GPT-5.6-sol		
Capability / task	Zero-shot	TSHarness	Δ	Zero-shot	TSHarness	Δ
CaTSBench						
Caption retrieval: cross-domain	100.00	100.00	0.00	100.00	100.00	0.00
Caption retrieval: perturbed	100.00	100.00	0.00	100.00	100.00	0.00
Caption retrieval: same-domain	100.00	100.00	0.00	100.00	100.00	0.00
Paraphrase consistency	100.00	100.00	0.00	100.00	100.00	0.00
Series comparison: amplitude	94.12	**100.00**	+5.88	100.00	100.00	0.00
Series comparison: mean	97.06	**100.00**	+2.94	100.00	100.00	0.00
Series comparison: peak timing	70.59	**91.18**	+20.59	94.12	**100.00**	+5.88
Series comparison: phenomenon	94.12	**100.00**	+5.88	97.06	97.06	0.00
Series comparison: volatility	76.47	**88.24**	+11.76	97.06	**100.00**	+2.94
Series retrieval: cross-domain	100.00	100.00	0.00	100.00	100.00	0.00
Series retrieval: perturbed	**82.35**	79.41	-2.94	**97.06**	94.12	-2.94
Series retrieval: same-domain	100.00	100.00	0.00	100.00	100.00	0.00
TSAQA						
Anomaly detection	49.28	49.28	0.00	44.93	44.93	0.00
Characterization	**84.06**	81.16	-2.90	59.42	**62.32**	+2.90
Classification	44.93	**50.72**	+5.80	50.72	50.72	0.00
Comparison	**81.16**	73.91	-7.25	**62.32**	59.42	-2.90
Data transformation	39.71	**55.88**	+16.18	69.12	**73.53**	+4.41
Temporal relationships	13.24	**45.59**	+32.35	60.29	60.29	0.00

Table 10: Accuracy before and after TPS evidence integration within the same harness trajectory. Initial Accuracy scores the recorded pre-TPS hypothesis; Final Accuracy scores the final prediction. Both are reported in percent. Δ is the net accuracy change in percentage points (pp), computed before rounding. Revisit gives the percentage of questions with an accepted evidence request. The initial hypothesis is a within-trajectory reference, distinct from the independently evaluated zero-shot baseline. Time-MQA uses the same subset as Table 8.

Benchmark	Initial Accuracy (%)	Final Accuracy (%)	Δ (pp)	Revisit (%)
Qwen3.8-27B				
TSRBench	44.17	62.62	+18.45	0.00
Time-MQA	59.47	65.78	+6.31	0.24
ChatTime	74.30	96.10	+21.80	0.00
CaTSBench	93.45	96.60	+3.16	0.00
TSAQA	52.18	59.47	+7.28	0.24
TimeOmni-1	51.21	55.10	+3.88	0.00
GPT-5.6-sol				
TSRBench	62.86	72.09	+9.22	19.90
Time-MQA	63.83	67.48	+3.64	2.18
ChatTime	89.78	96.88	+7.10	0.00
CaTSBench	99.03	99.27	+0.24	0.97
TSAQA	61.17	58.50	-2.67	11.41
TimeOmni-1	75.00	74.76	-0.24	48.54

Time-MQA, ChatTime, and CaTSBench, but its revisions on TSAQA and TimeOmni-1 yield small net losses. These comparisons characterize the complete evidence-integration process; they do not isolate the contribution of the TPS representation or additional loop iterations. Appendix B.5 gives concrete successful executions.

D.3 Efficiency-oriented execution across stages and capabilities

The original efficiency-oriented selector comparison uses Qwen3.8-27B on paired TSRBench questions with the resources, downstream procedures, and response-cache protocol specified in Appendix C. Controller executes the Perceiver's validated proposal without selection; RL applies the efficiency-oriented Tool Selector. These execution breakdowns do not describe the conservative TSHarness* configuration. All counts include executions across every input series, following Eq. (16).

Where execution savings occur. Figure 5 decomposes the reduction from 12.25 to 5.34 executions per question. Base perception accounts for the savings: its mean falls from 8.97 to 1.88. Question-specific tools remain at 0.62, while planned extensions increase from 2.67 to 2.84. Later evidence acquisition therefore remains active after a smaller initial tool set is executed. At the operator level, the largest reductions occur in statistics, value profiles, volatility, and change-point analysis. Trend and cross-series operations retain substantial execution volume, while question-specific verification tools retain their calls.

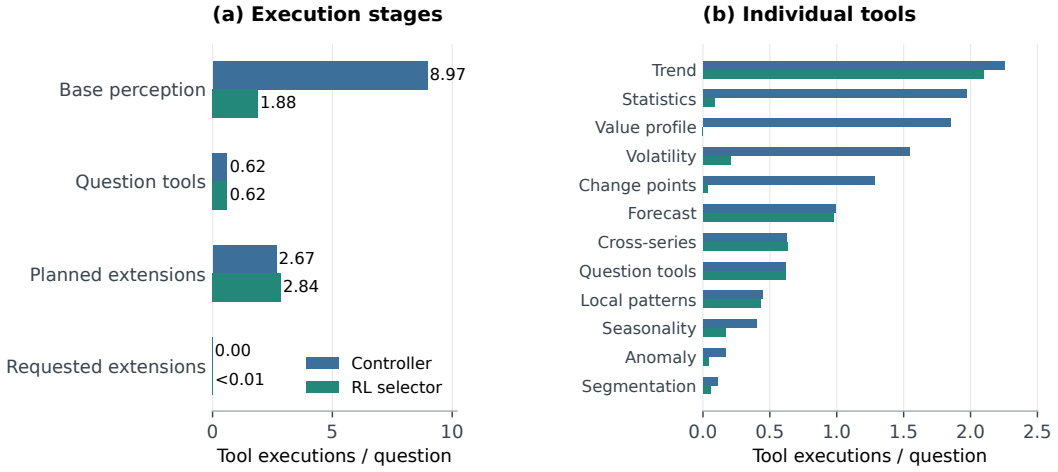

Figure 5: Tool execution breakdown for the matched Controller and RL conditions. (a) Mean executions at each stage. (b) Mean executions by tool across all stages, ordered by Controller usage; the seven question-specific analytical and verification tools are pooled as Question tools. Both panels use executions per question; requested extensions average zero for Controller and less than 0.01 for RL.

How the trade-off varies across capabilities. Figure 6 reports execution demand together with accuracy. Executions decrease in all four capability groups. Decision-making and reasoning retain their accuracies at 47.62% and 66.67%, respectively, with more than 55% fewer executions. Perception and prediction exhibit accuracy decreases of 4.29 and 4.63 percentage points. Within prediction, numerical forecasting improves from 45.83% to 50.00% while event forecasting decreases from 69.44% to 47.22%. The cost–quality balance therefore varies with the evidence demands of the task, even under the same selection policy.

Distribution and measurement scope. The reduction is broadly distributed: executions decrease on 411 questions and remain unchanged on one. The median falls from six to three, the interquartile range from 5–9 to 2–4, and the 95th percentile from 38 to 17. The 56.39% reduction in mean executions has a paired-bootstrap 95% interval of 55.41–57.39%, using 20,000 question-level resamples. Accuracy changes from 61.89% to 59.95%, a decrease of 1.94 percentage points. These statistics quantify an execution–accuracy trade-off under the fixed comparison. Accepted evidence revisits occur on

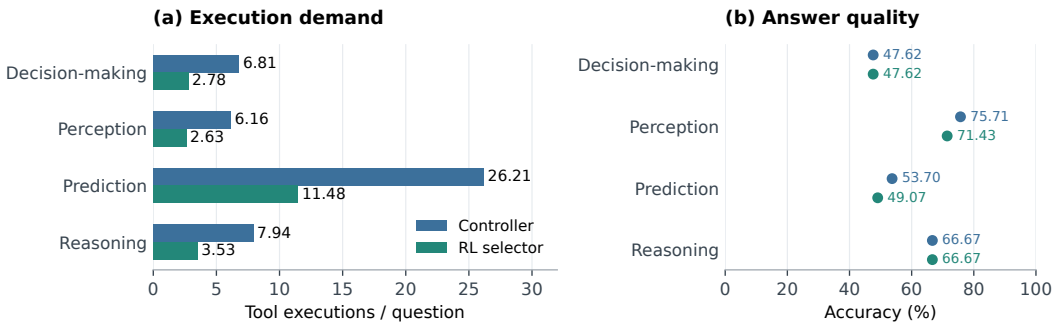

Figure 6: Execution–accuracy trade-off across TSRBench capabilities. (a) Mean tool executions per question across all stages. (b) Accuracy on the same questions. The selector reduces execution demand in every group while preserving accuracy in decision-making and reasoning. Colors identify the same policies in both panels.

zero Controller questions and one RL question; logical model-call totals are 3,393 and 3,392. These counts quantify the analytical work saved through tool selection; end-to-end runtime additionally depends on operation runtimes and LLM inference.

D.4 Selector operating points and transfer scope

Table 11 reports all five target benchmarks for the accuracy-oriented and efficiency-oriented configurations. Their mean accuracies are 74.71% and 72.54%, respectively, computed from the five benchmark scores. TSRBench is a source benchmark and is excluded from this target mean. The efficiency-oriented configuration additionally reaches 59.95% on TSRBench; this source-domain score does not enter Figure 4(a).

Table 11: Target accuracy (%) for the two deployment regimes. Accuracy-oriented values correspond to TSHarness* in Table 1. Both regimes reuse the same learned checkpoint.

Benchmark	Accuracy-oriented	Efficiency-oriented
Time-MQA	65.78	63.67
ChatTime	96.12	94.90
CaTSBench	96.84	93.69
TSAQA	59.71	57.04
TimeOmni	55.10	53.40
Five-target mean	74.71	72.54

Relative execution cost. For each matched comparison, the no-selector Controller condition defines 100% cost. Relative tool cost is the ratio of total executions, including every analysis stage and input series:

$$C = 100 \frac{\sum_{q \in \mathcal{Q}} N_{\text{all}}^{\text{selector}}(q)}{\sum_{q \in \mathcal{Q}} N_{\text{all}}^{\text{Controller}}(q)}. \tag{18}$$

Here $\mathcal{Q}$ contains the evaluated questions from all five target benchmarks. Costs are pooled over these questions, whereas accuracy is the unweighted mean of the five benchmark scores. Accuracy-oriented deployment retains 98.24% of the matched Controller's executions, including a 3.5% reduction on Time-MQA; efficiency-oriented deployment retains 43.61%. Table 12 gives all five measured configurations. Consistent with the execution penalty in Eq. (7), relative cost measures tool use; wall-clock latency and LLM inference are separate quantities.

The five configurations demonstrate that a fixed learned selector can support different accuracy–cost balances through its deployment rules. More conservative evidence acquisition sustains higher accuracy, while stronger filtering reduces tool execution. Inference rules and evidence processing differ across configurations, as discussed in Appendix C.3; the comparison evaluates complete deployment settings rather than an isolated selector parameter.

Table 12: Measured accuracy–cost trade-offs across all five target benchmarks, plotted in Figure 4(a). Tool savings equal 100% minus relative execution cost; each cost is normalized to its matched Controller.

Configuration	Accuracy (%)	Relative cost (%)	Tool savings (%)
Efficiency-oriented	72.54	43.61	56.39
Intermediate 1	73.62	57.43	42.57
Intermediate 2	74.10	74.28	25.72
Intermediate 3	74.55	85.23	14.77
Accuracy-oriented	74.71	98.24	1.76