TSQAgent: Rating Time Series Data Quality via Dedicated Agentic Reasoning

Shunyu Wu[1,2], **Dan Li**[1]*, **Haozheng Ye**[1,2], **Weibin Feng**[1,2], **Jian Lou**[1,2],
Bo Zhang[3], **Wenjie Feng**[4], **Chenjuan Guo**[5], **See-Kiong Ng**[6]
[1]Sun Yat-sen University
[2]State Key Laboratory of Internet Architecture [3]China University of Mining and Technology
[4]University of Science and Technology of China [5]East China Normal University
[6]National University of Singapore
wushy88@mail2.sysu.edu.cn, lidan263@mail.sysu.edu.cn

Abstract

Assessing the quality of time series (TS) data is fundamental yet inherently challenging due to the multifaceted nature of quality dimensions. Recently, large language models (LLMs) have emerged as a promising paradigm for TS quality assessment via pairwise comparison and per-dimension evaluation. However, existing approaches rely on manually predefined quality dimensions and purely text-based reasoning, leaving it unknown whether LLMs can identify truly relevant quality dimensions or perform grounded and quantitative quality comparisons. To investigate this, we construct TSQBench, a dedicated benchmark for evaluating LLMs on two progressive capabilities: (i) understanding and identifying relevant quality dimensions, and (ii) performing quality comparison under specific dimensions. Our analysis reveals that current LLMs consistently struggle with both dimension identification and evidence-grounded quality comparison. To address these limitations, we propose TSQAGENT, a novel agentic reasoning framework for TS quality rating consisting of three collaborative roles: *Perceiver* for focused dimension selection, *Inspector* for dimension-wise quantitative analysis, and *Adjudicator* that aggregates and refines the final judgment. In particular, we introduce an agentic reasoning strategy that instills the ability to identify and prioritize the most relevant quality dimensions, and further propose an agent workflow equipped with external analytical tools to enable precise quantitative comparisons over selected dimensions. Experiments on both the proposed benchmark and eleven real-world datasets demonstrate that our framework not only substantially improves LLMs' capabilities in quality understanding and quantitative comparison but also effectively translates these improvements into better quality-aware data selection, leading to enhanced downstream performance and data efficiency. Code is available at https://github.com/clsr1008/TSQAgent.

1 Introduction

The quality of time series (TS) data plays a critical role in building performant TS models, as low-quality data can substantially degrade model capability. In real-world scenarios, TS data is notoriously affected by various quality issues, including missing values, outliers, noise contamination, distribution shifts, and structural distortions (e.g., trend or frequency inconsistencies) (Fang et al., 2024; Zhang et al., 2024; Liu et al., 2024a). These issues render quality assessment of TS data inherently difficult, as it requires synthesizing multiple interacting quality dimensions and accurately

*Dan Li is the Corresponding Author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

assessing each dimension according to its specific evaluation criteria. Existing approaches primarily rely on statistical feature analysis (Jadidi et al., 2023) and heuristic rules (Rahm et al., 2000) to identify low-quality samples, or extend general data attribution (Influence Functions (Hampel, 1974) or Shapley (Shapley, 1953)) to TS quality (Zhang et al., 2025b; Bento et al., 2021). A more detailed discussion of related approaches is provided in Section A.

Recently, propelled by the success of large language models (LLMs) in a range of TS tasks from forecasting (Liu et al., 2024b; Gruver et al., 2023) to TS question answering (TSQA) (Kong et al., 2025; Guan et al., 2025; Langer et al., 2025), there has been growing interest in leveraging their intrinsic TS knowledge to enable comprehensive, multi-dimensional TS quality assessment. Along this direction, recent works have explored the use of LLMs for time series understanding and quality-related analysis. For example, TimeSeriesExam (Cai et al., 2024) constructs a benchmark for evaluating LLMs' understanding of temporal patterns (e.g., trends and periodicity) and quality-related signals, while AXIS (Lan et al., 2025) leverages LLMs to interpret anomalous patterns as potential indicators of data quality issues. These studies suggest that LLMs possess a certain level of temporal understanding and pave a promising foundation for time series quality assessment. Building upon this, TSRating (Wu et al., 2026a) explores LLM-based TS quality assessment by evaluating data along four predefined dimensions (namely trend, frequency, amplitude, and pattern).

Gap and Research Question. While such approaches establish a structured and automated assessment pipeline, they still rely on manually specified quality criteria and predominantly text-based reasoning, leaving the underlying capabilities of LLMs for TS quality assessment largely unexplored. This motivates us to investigate the following research question: *Can LLMs understand and adaptively identify relevant quality dimensions, and perform evidence-grounded quality comparisons under specific quality dimensions?*

Our Work. To investigate this question, we construct TSQBench, a dedicated benchmark to evaluate two capabilities of LLMs: quality dimension identification and evidence-grounded quality comparison. Specifically, we design a comprehensive set of quality dimensions, and leverage controlled quality degradation guided by semantic factors to generate paired time series with diverse and fine-grained quality differences (Xie et al., 2025). The benchmark results reveal two gaps in LLM-based TS quality assessment: (i) a lack of awareness of which quality dimensions are truly informative for selecting high-quality TS data (Zhang et al., 2026b), and (ii) an inherent weakness in reliable numerical reasoning over time series signals (Wu et al., 2026c), which often leads to unstable or hallucinated quality judgments.

Motivated by these findings, we propose TSQAGENT, a novel agentic reasoning framework for TS quality assessment and pairwise comparison. The agent system decomposes the task into three collaborative roles: (1) *Perceiver* builds a high-level understanding of the input pair and selects relevant quality dimensions for evaluation; (2) *Inspector* performs dimension-wise analysis through iterative reasoning with action-based evidence acquisition in a ReAct-style manner (Yao et al., 2023); and (3) *Adjudicator* aggregates all evidence into the final judgment while performing reflection to ensure consistency and completeness (Abbas et al., 2025; Li et al., 2025). In particular, we introduce an agentic reasoning strategy (Shao et al., 2024) that leverages TSQBench to train the Perceiver via tailored Group Relative Policy Optimization (GRPO) for adaptive and focused quality dimension selection, moving beyond fixed predefined criteria toward structured reasoning over heterogeneous quality factors. Next, we design a tool-augmented agent workflow that equips the Inspector with external analytical tools for grounded quantitative comparison over selected dimensions, where tool outputs are explicitly injected into the reasoning process as evidence (Wei et al., 2026; Zhang et al., 2026a). Our tool suite consists of 16 specific TS analytical functions covering statistical, spectral, and structural measurements.

We conduct experiments on both TSQBench and eleven real-world datasets, covering a full pipeline from pairwise quality judgments to training a dedicated quality rating model and downstream quality-aware data selection. The results demonstrate that our framework not only substantially improves LLMs' capabilities in quality understanding and quantitative comparison but also effectively translates these improvements into better data selection performance, leading to enhanced downstream accuracy and data efficiency. Notably, we fine-tune Timer-S1 (Liu et al., 2026), one of the largest publicly available TS foundation models with 8.3B parameters, and show that using only 75% of the data selected by our method achieves performance comparable to training on the full data pool.

Summary of Contributions. Our main contributions are summarized as follows:

- We identify and formalize the problem of TS quality assessment for LLMs, and construct a dedicated benchmark to evaluate two key capabilities of LLMs: TS quality dimension identification and evidence-grounded quality comparison.
- We propose TSQAGENT, an agentic reasoning framework that integrates reasoning-driven dimension selection and tool-augmented quantitative analysis to address the core challenges of TS quality assessment.
- We conduct extensive experiments demonstrating that our approach improves both LLM capabilities and downstream data selection performance across diverse datasets and models.

2 Measuring LLM Capabilities in Time Series Quality Rating

To systematically evaluate LLMs in time series quality understanding, we construct a synthetic benchmark, named TSQBench, based on pairwise comparison. TSQBench is designed to assess two core capabilities: (i) **quality dimension identification**, i.e., whether an LLM can correctly identify which quality dimensions (e.g., noise level or frequency patterns) two TS segments differ on; and (ii) **quality comparison**, i.e., whether an LLM can correctly decide which series exhibits higher quality under a given dimension.

Benchmark Design. TSQBench is built on a controlled synthetic generation pipeline, where paired time series are derived from a shared template and then injected with dimension-specific degradations. We define a unified quality dimension set $\mathcal{D} = \{d_1, \ldots, d_7\}$, including missing value, noise level, rare pattern, trend, frequency, amplitude, and pattern consistency, as detailed in Table 6. These dimensions describe data integrity, signal irregularities, and structural properties of temporal dynamics, providing a comprehensive basis for quality comparison (Cleveland et al., 1990; Hyndman & Athanasopoulos, 2018; Kazemi et al., 2019). Each benchmark instance activates a subset of dimensions, where asymmetric perturbations are applied to one side, enabling controlled and interpretable quality differences. Further details on TSQBench construction are provided in Appendix B.

Evaluation and Findings. We evaluate different LLMs on TSQBench using 1,000 synthetic samples, where Precision/Recall are used for dimension identification and accuracy is used for quality comparison. The results in Table 1 reveal two consistent limitations of existing LLMs in TS quality reasoning. For dimension identification, models exhibit low precision (33.4%–43.0%) despite relatively high recall (63.9%–83.0%), indicating unstable dimension selection behavior and a tendency to over-select irrelevant quality dimensions. For quality comparison, all models achieve only marginal improvements over random guessing (50%), with accuracies ranging from 54.7% to 61.0%, suggesting limited intrinsic capability in performing reliable fine-grained quality comparison. These observations motivate the two core designs in TSQAgent proposed in Section 3: (i) a reasoning-oriented dimension selection mechanism trained via TS quality-tailored GRPO, and (ii) a tool-augmented agentic reasoning for quantitative quality comparison.

Table 1: Performance of LLM-based TS quality assessment on TSQBench. We evaluate two core capabilities: dimension identification and quality comparison.

Model	Qwen3-4B	Gemma4-E4B	Phi4-mini	Qwen3-8B	GPT-4o mini	Claude Haiku 4.5
			Dimension Identification			
Precision	33.5%	38.0%	33.4%	37.4%	34.7%	43.0%
Recall	80.9%	75.7%	69.7%	78.0%	83.0%	63.9%
			Quality Comparison			
Accuracy	54.7%	58.6%	54.9%	58.4%	60.0%	61.0%

3 TSQAgent: Agentic Reasoning Framework Tailored for TS Quality

3.1 Overview of TSQAgent

TSQAgent is a multi-agent framework for pairwise time series quality assessment, designed to determine which of two time series exhibits higher overall quality through structured reasoning

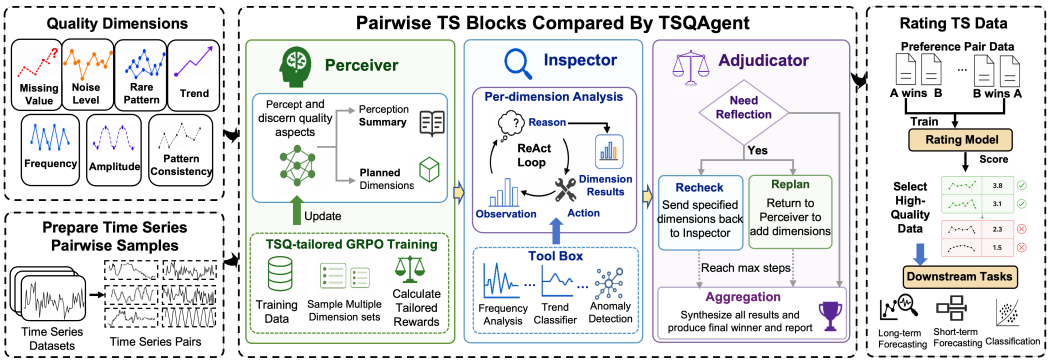


Figure 1: Overview of the proposed TSQAgent framework for time series quality rating.

and tool-grounded analysis. The pairwise formulation simplifies evaluation via relative comparison, avoiding absolute scoring and improving robustness across diverse scenarios (Peng et al., 2025). The system decomposes the task into three specialized agents that operate collaboratively: (1) **Perceiver**: constructs a high-level understanding of the input pair and selects relevant quality dimensions for evaluation; (2) **Inspector**: performs dimension-wise analysis via iterative reasoning and tool-based evidence acquisition; and (3) **Adjudicator**: aggregates dimension-level evidence into a final judgment while performing reflection to ensure consistency and completeness. The overall architecture of TSQAgent is shown in Figure 1.

Formally, given two time series segments $(\text{series}_A, \text{series}_B)$ and dataset context c, the framework produces a final comparative result $R = (w, s, C)$, where w denotes the higher-quality series, s the confidence level, and C a structured explanation grounded in multi-dimensional evidence. Specifically, the confidence level $s \in [0, 1]$ measures the reliability of each pairwise judgment based on the consistency and sufficiency of the available evidence.

To complement the data rating and selection process, we convert pairwise preferences into scalar quality scores. Specifically, TSQAgent first produces preference judgments between time series pairs. We then filter out low-confidence comparisons and train a lightweight rating model on these comparisons to assign a scalar quality score to each sample via Bradley–Terry optimization, thereby enabling global ranking and quality-aware data selection. The rating model architecture and training details are provided in Appendix E.

3.2 Perceiver: Perception and Planning

The Perceiver serves as the entry point of TSQAgent, responsible for constructing a high-level understanding of the input pair $(\text{series}_A, \text{series}_B)$ and selecting relevant quality dimensions for subsequent analysis. Formally, the Perceiver can be viewed as a mapping

$$(Z, \mathcal{D}_{\text{plan}}) = \mathcal{A}_p(c, \text{series}_A, \text{series}_B), \tag{1}$$

where c denotes the dataset description, and the output consists of a perception summary Z, and a subset of selected quality dimensions $\mathcal{D}_{\text{plan}}$ for subsequent inspection.

Given the raw time series inputs and associated statistical summaries (e.g., mean, standard deviation, quantiles), the Perceiver constructs a perception summary Z that captures the salient differences between the two series. This reasoning process is guided by a predefined set of quality dimensions $\mathcal{D} = \{d_1, \ldots, d_7\}$ (see Table 6), which characterizes key quality aspects such as data completeness, noise characteristics, and temporal structure. Based on this, the Perceiver selects a subset of relevant dimensions $\mathcal{D}_{\text{plan}} \subseteq \mathcal{D}$, emphasizing the most informative aspects for subsequent analysis.

Agentic Reasoning for TS Quality Dimension Selection. Importantly, we formulate time series quality dimension selection as an agentic reasoning process within the Perceiver, where the agent is required to infer and prioritize the most decision-relevant quality aspects for each input pair. To instill this capability, we introduce a TS quality-tailored GRPO (Group Relative Policy Optimization) training strategy (Shao et al., 2024), which serves as the underlying optimization framework. Our training dataset consists of 3,320 samples derived from TSQBench pipeline (see Section 2), and the optimization is guided by a structured reward that enforces both output validity and alignment

between predicted and ground-truth dimension sets. This induces more discriminative reasoning over time series quality dimensions, improving fine-grained differentiation across diverse series pairs. More implementation details including training configurations are provided in Appendix C.

3.3 Inspector: Dimension-wise Analysis

The Inspector is responsible for fine-grained evaluation of each selected quality dimension and serves as the core reasoning module in TSQAgent. Given the planned dimensions $\mathcal{D}_{\text{plan}}$ and perception summary Z, it performs pairwise comparison between both series, producing structured judgments for each dimension.

The Inspector executes evaluation in a dimension-by-dimension manner over $\mathcal{D}_{\text{plan}}$, performing pairwise assessment between series_A and series_B under a bounded ReAct-style interaction (Yao et al., 2023). Formally, for each dimension d, the process proceeds through an iterative reasoning-action loop:

$$q_d \rightarrow (\text{Reason}_t \rightarrow \text{Action}_t \rightarrow \text{Observation}_t)^{*_K} \rightarrow R_d. \tag{2}$$

Here, q_d denotes the evaluation query derived from the input pair $(\text{series}_A, \text{series}_B)$ and the specification of dimension d. Each iteration consists of three components: Reason_t performs internal reasoning over observed signals and evidence; Action_t optionally invokes external analytical tools to obtain quantitative evidence; and Observation_t captures the corresponding tool outputs or retrieved signals. $R_d = (w_d, s_d, E_d, C_d)$ represents the final assessment, including the predicted winner $w_d \in \{A, B, \text{TIE}\}$, confidence score $s_d \in [0, 1]$, supporting evidence E_d, and textual conclusion C_d. K denotes the maximum number of reasoning-action iterations per dimension, ensuring that each dimension is evaluated within a limited computational budget, after which the Inspector finalizes the outputs based on accumulated evidence.

Tool-Augmented Quantitative Quality Comparison. Within each ReAct loop, the Inspector performs dimension-specific reasoning over raw time series, leveraging internal knowledge and accumulated evidence to identify salient patterns relevant to the target dimension, such as trend shifts, periodic structures, or rare behaviors. When internal reasoning is insufficient to resolve subtle differences, the Inspector invokes external analytical tools to obtain quantitative evidence (Qian et al., 2025; Schick et al., 2023). In contrast to purely qualitative reasoning, tool-based signals provide measurable and fine-grained evidence that is often difficult to infer directly from raw sequences, thereby improving decision robustness (Wu et al., 2026c). To support this process, we develop a library of analytical tools for data completeness, noise, rare pattern detection, and temporal structure analysis, as detailed in Appendix D. Tool outputs are treated as structured observations within the reasoning trajectory, and are cached for reuse across dimensions to reduce redundant computation.

3.4 Adjudicator: Reflection and Aggregation

The Adjudicator is responsible for producing a final comparative judgment between the two time series by aggregating dimension-level assessment results from the Inspector, while also ensuring the completeness and consistency of the evidence through a reflection mechanism.

Formally, given the dimension-level results $\mathcal{R} = \{R_d \mid d \in \mathcal{D}_{\text{plan}}\}$ with $R_d = (w_d, s_d, E_d, C_d)$ produced by the Inspector, the Adjudicator first performs a reflection step over $\mathcal{R}$ to evaluate whether the available evidence is sufficient and mutually consistent. This process yields a reflection directive $\rho \in \{\texttt{recheck}(\mathcal{D}_{\text{recheck}}), \texttt{replan}, \varnothing\}$, where $\varnothing$ indicates that no further refinement is required.

The reflection directive distinguishes two types of refinement. A $\texttt{recheck}(\mathcal{D}_{\text{recheck}})$ signal is generated when the evidence for certain dimensions is insufficient or internally conflicting, where $\mathcal{D}_{\text{recheck}} \subseteq \mathcal{D}_{\text{plan}}$ specifies the subset of dimensions to be re-evaluated by the Inspector. Alternatively, a `replan` signal is issued when the current dimension set $\mathcal{D}_{\text{plan}}$ is incomplete with respect to important aspects of data quality, requiring the Perceiver to introduce additional dimensions for further assessment. These reflection signals enable iterative refinement of the evaluation process across agents (Shinn et al., 2023). To ensure stable execution, reflection steps are explicitly bounded. When the budget is exhausted, the reflection process terminates and is treated as $\rho = \varnothing$, after which the Adjudicator proceeds to aggregate all dimension-level results, thereby preventing indefinite refinement loops.

When the evidence is considered sufficient and consistent ($\rho = \varnothing$), the Adjudicator aggregates the dimension-level results by considering the predicted winners, confidence scores, supporting evidence, and textual conclusions. It then produces the final judgment $R = (w, s, C)$. Further analysis of dimension-level interpretability and task-dependent quality assessment is provided in Appendix F.7.

4 Experiments

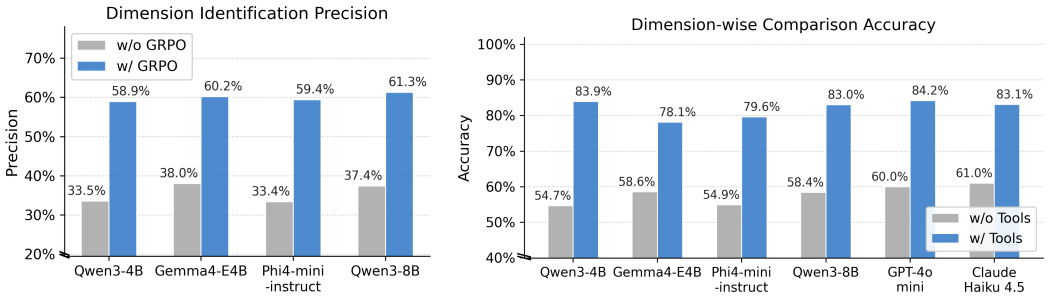
We evaluate TSQAgent from two complementary perspectives: (1) controlled evaluation on TSQBench to verify how reasoning-driven dimension selection and tool augmentation improve the LLMs' capabilities of TS quality evaluation; (2) real-world data selection experiments to evaluate the effectiveness of TSQAgent across diverse datasets and downstream models. Additional qualitative case studies are provided in Appendix G to illustrate the reasoning behaviors of the proposed agents.

4.1 Capability Evaluation on TSQBench

We conduct capability evaluation on TSQBench to separately assess the two core capabilities of LLMs: dimension identification and dimension-wise quality comparison. We evaluate a diverse set of both open-source and proprietary models, including Qwen3 (Yang et al., 2025a), Gemma4 (Team, 2025), Phi4 (Abdin et al., 2024), GPT-4o mini (Achiam et al., 2023), and Claude Haiku 4.5. To examine whether the capabilities learned from TSQBench generalize beyond synthetic quality degradations, we provide an additional evaluation on real-world anomalies in Appendix F.6.

Dimension Identification Precision. As shown in Figure 2a, we evaluate the effect of our TS quality-tailored GRPO training strategy on dimension selection performance. The strategy consistently improves dimension identification precision across all evaluated open-source models, increasing precision from around 33%–38% to nearly 60%. These results indicate that vanilla LLMs tend to over-select irrelevant quality dimensions, while our reasoning-based training strategy effectively encourages more focused and discriminative dimension selection behavior. Additional results using the F1 score are provided in Appendix F.3.

Quantitative Comparison Accuracy. Figure 2b reports the performance of models with and without tool access for quality comparison. Tool augmentation consistently improves comparison accuracy across all evaluated models, raising performance from approximately 55%–61% to over 78%–84%. The improvements are observed for both open-source and proprietary models, suggesting that external analytical tools provide critical quantitative evidence that is difficult to obtain through purely text-based reasoning alone.



(a) Effect of GRPO on Dimension Identification (b) Effect of Tool Augmentation on Quality Comparison

Figure 2: Capability Evaluation on TSQBench. Subplot (a) reports dimension identification performance under different models and training configurations, highlighting the effect of reasoning-driven dimension selection. Subplot (b) reports quantitative comparison performance with and without tool usage, showing the impact of external analytical support on fine-grained quality assessment.

4.2 Real-World Data Selection Experiments

The goal of this section is to evaluate whether TSQAgent can reliably assess data quality in real-world settings, and whether such assessments translate into improved downstream performance and data efficiency. By selecting high-quality subsets based on the TSQAgent's judgments, we test

whether comparable or superior performance can be achieved with reduced training data, thereby complementing the results on the synthetic TSQBench.

Datasets and Baselines. We conduct experiments on 11 widely-used real-world datasets across three task categories. For long-term forecasting, we use *Electricity* (Trindade, 2015), *ExchangeRate* (Lai et al., 2018), *Traffic* (Lai et al., 2018), and *Weather* (Wu et al., 2021). For short-term forecasting, we adopt the *M4 dataset* (Makridakis et al., 2018), including its *Yearly*, *Monthly*, and *Daily* subsets. For classification, we use four datasets from the UCR/UEA archive (Ismail Fawaz et al., 2019): *MedicalImages*, *CBF*, *BME*, and *Handwriting*. We compare our method with several representative data valuation and selection baselines, including DataShapley (Ghorbani & Zou, 2019), DataOob (Kwon & Zou, 2023), TimeInf (Zhang et al., 2025b), and TSRating (Wu et al., 2026a). All baselines produce sample-level scores. Details of the datasets and baselines are provided in the Appendix F.1.

Setups and Evaluation. For all tasks, we construct training, validation, and test splits with a ratio of 7:2:1 and then segment time series into fixed-length samples according to the requirements of each dataset (e.g., historical context length determined by the forecasting configuration). Each resulting sample is assigned a quality score by different methods, and we select the top 50% of samples to train downstream models (e.g., CNN, PatchTST (Nie et al., 2023)). The trained models are then evaluated on the held-out test set under standard metrics. We adopt task-specific metrics: Root Mean Squared Error (RMSE) for long-term forecasting, Mean Absolute Percentage Error (MAPE) for short-term forecasting, and Accuracy for classification. Better data selection is reflected by improved downstream performance. All results are averaged over 10 runs with different random seeds.

Table 2: Comparison of data selection methods under a 50% high-quality data budget across forecasting and classification tasks on multiple datasets and models. Best results are **bolded**.

Model	Method	Long-term (RMSE ↓)				Short-term (MAPE ↓)			Classification (Accuracy ↑)			
		Elec.	ExRate	Traffic	Wea.	M4Y	M4M	M4D	MImg	CBF	BME	HW
Linear	Random	1.601	0.356	0.979	0.665	1.705	1.208	1.672	0.390	0.294	0.427	0.053
	DataOob	1.539	0.318	0.761	0.638	1.949	1.133	1.779	0.457	0.318	0.260	0.028
	DataShapley	1.580	0.323	0.956	0.638	**1.488**	1.207	1.370	0.432	0.337	0.433	0.038
	TimeInf	1.391	0.272	**0.609**	0.616	1.966	1.178	1.463	0.428	0.320	0.500	0.042
	TSRating	1.390	0.275	0.683	0.611	1.577	1.112	1.409	0.459	0.361	0.507	0.049
	TSQAgent	**1.273**	**0.270**	0.611	**0.593**	1.494	**1.072**	**1.352**	**0.488**	**0.379**	**0.520**	**0.056**
CNN	Random	1.592	1.474	0.504	0.769	2.332	1.124	1.193	0.554	0.595	0.495	0.151
	DataOob	1.609	1.527	0.552	0.737	2.957	1.208	1.346	0.514	0.663	0.521	0.155
	DataShapley	1.529	1.598	0.475	0.767	2.553	1.117	1.159	0.550	0.593	0.564	0.159
	TimeInf	1.530	1.515	0.505	0.758	2.289	1.117	1.103	0.567	0.563	0.535	0.155
	TSRating	1.511	1.429	0.428	0.734	1.782	**1.075**	1.108	0.561	**0.679**	0.575	0.159
	TSQAgent	**1.419**	**1.371**	**0.412**	**0.723**	**1.744**	1.108	**1.096**	**0.574**	0.668	**0.580**	**0.162**
PatchTST	Random	0.406	0.222	0.361	0.474	4.146	2.044	1.985	0.561	0.470	0.475	0.124
	DataOob	0.416	0.226	0.362	0.558	3.982	2.060	1.958	0.571	0.489	0.489	0.141
	DataShapley	**0.396**	**0.212**	0.361	0.457	4.273	2.017	2.028	0.571	0.472	0.529	0.126
	TimeInf	0.415	0.220	0.351	0.510	4.029	2.019	2.023	0.522	0.467	0.504	0.131
	TSRating	0.397	0.213	0.351	0.467	3.901	1.957	1.863	0.572	0.511	**0.536**	0.156
	TSQAgent	0.405	0.214	**0.348**	**0.450**	**3.820**	**1.856**	**1.855**	**0.573**	**0.516**	0.533	**0.166**

4.2.1 Main Results

Table 2 presents the performance of different data selection methods across three downstream time series tasks with three representative downstream model architectures: Linear, CNN-based, and Transformer-based (PatchTST). Overall, our method consistently achieves the best or near-best performance across most settings, demonstrating strong effectiveness in identifying high-quality data. In long-term forecasting, it attains the lowest RMSE in the majority of cases, particularly showing clear advantages under Linear and CNN models. For short-term forecasting, our method obtains the best MAPE in 7 out of 9 cases, indicating robust performance across different temporal granularities. Likewise, in classification tasks, it ranks first in 10 out of 12 cases, suggesting that the selected subsets effectively preserve discriminative information. Compared with TSRating and TimeInf, our method shows more consistent gains across settings, indicating more accurate and effective quality assessment for guiding data selection. The results with standard deviations across random seeds are reported in Appendix F.2.

4.2.2 Component Analysis

Effect of Reasoning-driven Dimension Selection. We evaluate the impact of TS quality-tailored GRPO on Perceiver, with results summarized in Figure 3. The left plot reports downstream long-term forecasting performance (RMSE) across four datasets using PatchTST, while the right plot shows the averaged efficiency metrics, including selected dimensions, token usage, and inference time. Overall, our training strategy consistently improves data selection quality, leading to lower RMSE across all datasets. Meanwhile, it significantly reduces the number of selected dimensions as well as computational overhead in terms of token usage and inference latency. This suggests that the training strategy helps Perceiver focus on more informative dimensions while suppressing spurious or redundant reasoning paths, resulting in both more accurate and more efficient quality assessment.

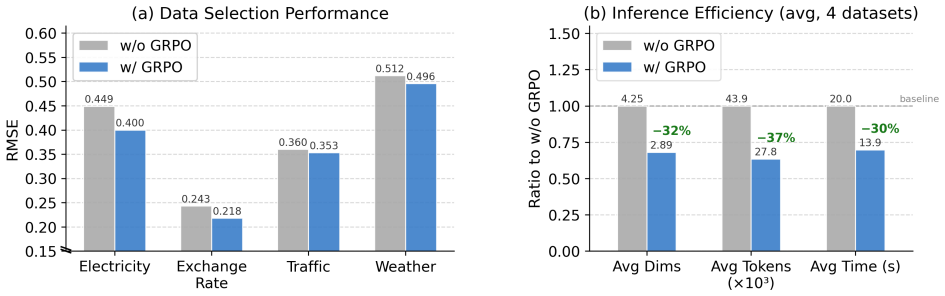


Figure 3: Effect of reasoning-driven Perceiver training. The left figure shows downstream data selection performance (RMSE) across four datasets using PatchTST as the downstream model, while the right figure reports efficiency metrics averaged over four datasets, including selected dimensions, token usage, and inference time.

Effect of Tool-Augmented Quantitative Comparison. We further study the impact of tool augmentation in the Inspector module, which enables explicit quantitative comparison, versus pure LLM-based reasoning without tool access. As shown in Figure 4, we evaluate three different LLM backbones, i.e., Qwen3-4B (Yang et al., 2025a), Gemma-3-4B (Team, 2025), and GPT-4o mini (Achiam et al., 2023), as the backbone of the TSQAgent across four datasets. The Inspector agent is required to perform data quality assessment under both settings, while downstream performance is measured by RMSE using PatchTST. The results indicate that enabling tool-based quantitative comparison generally improves performance across the majority of LLMs and datasets, suggesting that explicit quantitative signals enhance the reliability of data quality judgments and are broadly effective across different LLM backbones.

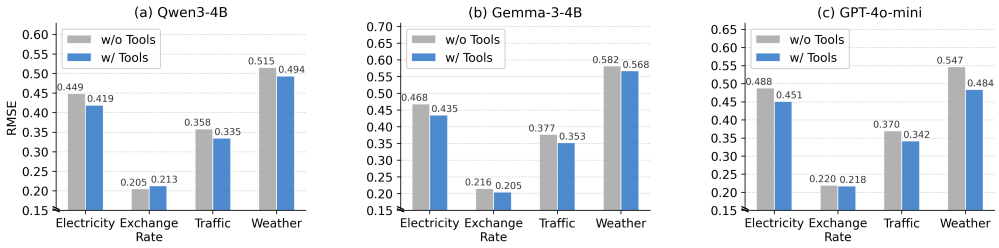


Figure 4: Effect of tool-augmented quantitative comparison across different LLMs, where each subplot corresponds to a specific LLM backbone. The figures report downstream data selection performance (RMSE) under two settings: with tool usage (Inspector performs explicit quantitative comparison) and without tool usage (pure text-based reasoning). Results are evaluated on four datasets using PatchTST as the downstream model.

Effect of Quality Dimension Design. We further study the impact of quality dimension design as summarized in Table 6. Specifically, we group the seven dimensions into three categories: *Data Quality* (Missing Value, Noise Level), *Rare Pattern* (Rare Pattern), and *Pattern Structure* (Trend, Frequency, Amplitude, Pattern Consistency). We construct three ablated variants by removing each group in turn and compare them with the full configuration. As shown in Figure 5, removing any

group consistently leads to performance degradation in most cases. In particular, excluding *Pattern Structure* or *Rare Pattern* results in the largest drops, indicating the importance of capturing temporal structures and informative irregularities. Removing *Data Quality* has a relatively smaller impact since such datasets exhibit limited variation in basic defects such as missing values and noise, making such signals less discriminative. Nevertheless, the full configuration achieves the best average performance, indicating that these dimension groups contribute complementary signals for reliable TS data quality assessment. The results with standard deviations for the component analysis experiments are reported in Appendix F.2.

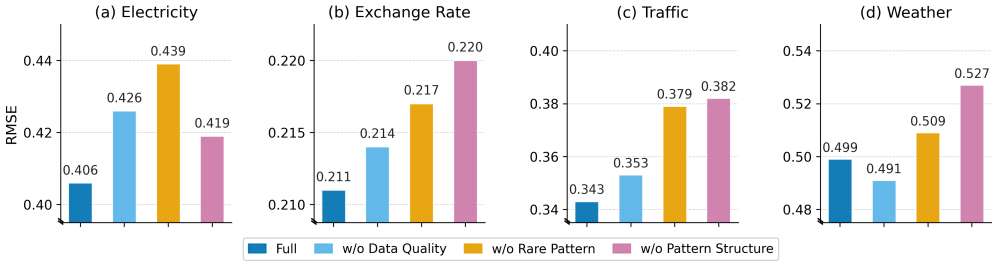


Figure 5: Effect of quality dimension design. Each subplot shows results on a specific dataset, comparing downstream data selection performance (RMSE) under different dimension configurations, including the full set of seven dimensions and three ablated variants with one dimension group removed (Data Quality, Rare Pattern, or Pattern Structure). Results are evaluated on four datasets using PatchTST as the downstream model.

Table 3: Cross-dataset data selection for TSFM (Timer-S1) fine-tuning. We compare random selection and quality-based selection under fixed data budgets, along with full-data training. Results are reported in RMSE (lower is better).

Config	Electricity	ExchangeRate	Traffic	Weather	Avg
Untrained	0.477	0.228	0.293	0.510	0.377
Random (50% data)	0.448	0.225	0.290	0.492	0.364
TSQAgent (50% data)	0.425	0.221	0.283	0.484	0.353
Random (75% data)	0.421	0.222	0.285	0.470	0.350
TSQAgent (75% data)	0.400	0.218	0.278	0.468	0.341
Full (100% data)	0.395	0.218	0.274	0.465	0.338

4.2.3 Cross-Dataset Selection for TSFM Fine-tuning

We further evaluate our method in a more challenging setting where data from multiple datasets are mixed for time series foundation model (TSFM) fine-tuning. The goal is to examine whether quality-aware selection remains effective under heterogeneous data distributions, and whether it can improve fine-tuning efficiency with limited data. Specifically, we construct a unified pool by merging samples from four long-term forecasting training sets. To ensure cross-dataset comparability, we train the rating model via meta-learning to produce globally consistent quality scores, with details provided in Appendix E.4. We further examine how meta-training diversity affects the generalization of the rating model in Appendix F.5. Based on these scores, we select the top 50% of samples in the pool for fine-tuning Timer-S1 (Liu et al., 2026), one of the largest publicly available TS foundation models, with 8.3B total parameters. The model is then evaluated on the test set of each dataset separately using RMSE. We compare against random selection with the same budget and full-data training. Additional comparisons with TSRating and DataShapley under the 50% data budget are provided in Appendix F.4.

Table 3 shows that our method consistently outperforms random selection across all datasets under both 50% and 75% data budgets, achieving lower RMSE with the same amount of training data. More importantly, when increasing the data budget to 75%, the performance of our method becomes very close to full-data training (0.341 vs. 0.338 on average). This indicates that our approach can effectively identify high-quality training samples, enabling near-full-data performance with reduced data usage, thereby improving the efficiency of TSFM fine-tuning.

4.2.4 Computational Cost and Efficiency

We further analyze the computational and API costs of TSQAgent under the cross-dataset TSFM selection setting described in Section 4.2.3. The unified training pool consists of four forecasting datasets (Electricity, ExchangeRate, Traffic, and Weather), containing 10,692 training samples in total. We collect 2,000 pairwise quality judgments through TSQAgent to train the shared meta rater.

Computational Cost. Table 4 reports the wall-clock time and amortized time per sample of TSQAgent and traditional data valuation baselines. TSQAgent achieves a favorable efficiency trade-off compared with traditional data valuation baselines, with an overall computational cost comparable to or lower than the comparison methods. Besides, the majority of TSQAgent's computational cost comes from the offline annotation stage, where the agentic framework is used to collect pairwise quality judgments. These judgments are then used to meta-train a shared rater across multiple datasets, enabling transferable quality assessment. Therefore, this upfront annotation and meta-training cost is a one-time expense, since the resulting meta rater can be efficiently adapted and reused.

At the inference stage, the trained rater model can be directly applied to new samples or datasets with only lightweight few-shot adaptation and rater inference costs. This stage requires only 64s in total (0.006s/sample), enabling efficient scaling to additional data. In comparison, although traditional data valuation methods do not require LLM-based annotation, they need to repeatedly compute sample importance for newly encountered samples or datasets, resulting in much higher amortized costs (e.g., 3.66s/sample for DataOob).

Table 4: Computational cost comparison under the cross-dataset TSFM fine-tuning setting.

Method / Component	Wall-clock Time (s)	Amortized Time (s/sample)
DataShapley	94197	8.81
DataOob	39101	3.66
TimeInf	28857	2.70
TSQAgent (Total)	**24548**	**2.30**
– Pairwise judgment annotation	24350	2.28
– Meta-training	134	0.013
– Few-shot adaptation + inference	64	0.006

API Cost. We additionally report the API cost of collecting the 2,000 pairwise judgments with different proprietary LLM backbones. The reported cost covers the complete TSQAgent pipeline, including the Perceiver, Inspector, and Adjudicator, together with external Python tool calls and ReAct-style reasoning loops. As summarized in Table 5, the annotation process consumes approximately 5.8–6.4 million input tokens and 0.26–0.35 million output tokens, with the resulting API cost depending on the selected LLM backbone.

Table 5: Token consumption and API cost for collecting 2,000 pairwise quality judgments using different proprietary LLM backbones.

LLM	Input Tokens	Output Tokens	API Cost
GPT-4o mini	6.05M	0.26M	$1.08
Gemini 2.5 Flash	5.82M	0.28M	$2.55
Claude Haiku 4.5	6.44M	0.35M	$7.50

5 Conclusion

In this paper, we investigate the capability of LLMs for time series quality assessment by constructing a dedicated benchmark TSQBench and identifying two key limitations: unreliable quality dimension identification and weak quantitative comparison ability. To address these limitations, we propose TSQAgent, an agentic reasoning framework that devises reasoning-oriented dimension selection with tool-augmented quantitative analysis for reliable TS quality assessment. Experiments on TSQBench and eleven real-world datasets demonstrate that TSQAgent consistently improves TS quality reasoning and enables more effective quality-aware data selection. Future work will address limitations related to generalization beyond synthetic quality degradations, predefined quality dimensions, agentic inference efficiency, and scalability to broader application settings, as detailed in Appendix H.

Acknowledgments and Disclosure of Funding

We would like to thank anonymous reviewers and area chairs for their constructive feedback and efforts in improving our paper. This work was supported by Tencent Rhino-Bird Research Fund (JR2026TEG604). This work was supported by the Open Research Fund of State Key Laboratory of Internet Architecture. We also gratefully acknowledge the computational resources and support provided by the National Supercomputer Center in Guangzhou.

References

Adnan Abbas, Caleb Wohn, Donghan Hu, Eugenia H Rho, and Sang Won Lee. Pitch: designing agentic conversational support for planning and self-reflection. In *Proceedings of the 7th ACM Conference on Conversational User Interfaces*, pp. 1–22, 2025.

Marah Abdin, Jyoti Aneja, Harkirat Behl, Sébastien Bubeck, Ronen Eldan, Suriya Gunasekar, Michael Harrison, Russell J Hewett, Mojan Javaheripi, Piero Kauffmann, et al. Phi-4 technical report. *arXiv preprint arXiv:2412.08905*, 2024.

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Chuadhry Mujeeb Ahmed, Venkata Reddy Palleti, and Aditya P Mathur. Wadi: A water distribution testbed for research in the design of secure cyber physical systems. In *Proceedings of the 3rd international workshop on cyber-physical systems for smart water networks*, pp. 25–28, 2017.

Julien Audibert, Pietro Michiardi, Frédéric Guyard, Sébastien Marti, and Maria A Zuluaga. Usad: Unsupervised anomaly detection on multivariate time series. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 3395–3404, 2020.

Anthony Bagnall, Hoang Anh Dau, Jason Lines, Michael Flynn, James Large, Aaron Bostrom, Paul Southam, and Eamonn Keogh. The uea multivariate time series classification archive, 2018. *arXiv preprint arXiv:1811.00075*, 2018.

João Bento, Pedro Saleiro, André F Cruz, Mário AT Figueiredo, and Pedro Bizarro. Timeshap: Explaining recurrent models through sequence perturbations. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, pp. 2565–2573, 2021.

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

Mingyue Cheng, Xiaoyu Tao, Qi Liu, Ze Guo, and Enhong Chen. Position: Beyond model-centric prediction–agentic time series forecasting. *arXiv preprint arXiv:2602.01776*, 2026.

Qisen Cheng, Jinming Xing, Chang Xue, and Xiaoran Yang. Unifying prediction and explanation in time-series transformers via shapley-based pretraining. In *2025 21st IEEE International Colloquium on Signal Processing & Its Applications (CSPA)*, pp. 325–330. IEEE, 2025.

James Clark, Zhen Liu, and Nathalie Japkowicz. Adaptive threshold for outlier detection on data streams. In *2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA)*, pp. 41–49. IEEE, 2018.

Robert B Cleveland, William S Cleveland, Jean E McRae, Irma Terpenning, et al. Stl: A seasonal-trend decomposition. *J. off. Stat*, 6(1):3–73, 1990.

Hoang Anh Dau, Anthony Bagnall, Kaveh Kamgar, Chin-Chia Michael Yeh, Yan Zhu, Shaghayegh Gharghabi, Chotirat Ann Ratanamahatana, and Eamonn Keogh. The ucr time series archive. *IEEE/CAA Journal of Automatica Sinica*, 6(6):1293–1305, 2019.

Shikai Fang, Qingsong Wen, Yingtao Luo, Shandian Zhe, and Liang Sun. Bayotide: Bayesian online multivariate time series imputation with functional decomposition. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 12993–13009, 2024.

Azul Garza and Renée Rosillo. Timecopilot. *arXiv preprint arXiv:2509.00616*, 2025.

Amirata Ghorbani and James Zou. Data shapley: Equitable valuation of data for machine learning. In *International conference on machine learning*, pp. 2242–2251. PMLR, 2019.

Jonathan Goh, Sridhar Adepu, Khurum Nazir Junejo, and Aditya Mathur. A dataset to support research in the design of secure water treatment systems. In *International conference on critical information infrastructures security*, pp. 88–99. Springer, 2016.

Peiliang Gong, Emadeldeen Eldele, Min Wu, Zhenghua Chen, Xiaoli Li, and Daoqiang Zhang. Bridging distribution gaps in time series foundation model pretraining with prototype-guided normalization. *IEEE Transactions on Neural Networks and Learning Systems*, 2026.

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

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

Tong Guan, Zijie Meng, Dianqi Li, Shiyu Wang, Chao-Han Huck Yang, Qingsong Wen, Zuozhu Liu, Sabato Marco Siniscalchi, Ming Jin, and Shirui Pan. Timeomni-1: Incentivizing complex reasoning with time series in large language models. *arXiv preprint arXiv:2509.24803*, 2025.

Frank R Hampel. The influence curve and its role in robust estimation. *Journal of the american statistical association*, 69(346):383–393, 1974.

Kyle Hundman, Valentino Constantinou, Christopher Laporte, Ian Colwell, and Tom Soderstrom. Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 387–395, 2018.

Rob J Hyndman and George Athanasopoulos. *Forecasting: principles and practice*. OTexts, 2018.

Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. Deep learning for time series classification: a review. *Data mining and knowledge discovery*, 33(4):917–963, 2019.

Zahra Jadidi, Shantanu Pal, Mukhtar Hussain, and Kien Nguyen Thanh. Correlation-based anomaly detection in industrial control systems. *Sensors*, 23(3):1561, 2023.

Yushan Jiang, Wenchao Yu, Geon Lee, Dongjin Song, Kijung Shin, Wei Cheng, Yanchi Liu, and Haifeng Chen. Timexl: Explainable multi-modal time series prediction with llm-in-the-loop. *arXiv preprint arXiv:2503.01013*, 2025.

Seyed Mehran Kazemi, Rishab Goel, Sepehr Eghbali, Janahan Ramanan, Jaspreet Sahota, Sanjay Thakur, Stella Wu, Cathal Smyth, Pascal Poupart, and Marcus Brubaker. Time2vec: Learning a vector representation of time. *arXiv preprint arXiv:1907.05321*, 2019.

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, 2025.

Yongchan Kwon and James Zou. Data-oob: Out-of-bag estimate as a simple and efficient data value. In *International conference on machine learning*, pp. 18135–18152. PMLR, 2023.

Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. Modeling long-and short-term temporal patterns with deep neural networks. In *The 41st international ACM SIGIR conference on research & development in information retrieval*, pp. 95–104, 2018.

Tian Lan, Hao Duong Le, Jinbo Li, Wenjun He, Meng Wang, Chenghao Liu, and Chen Zhang. Axis: Explainable time series anomaly detection with large language models. *arXiv preprint arXiv:2509.24378*, 2025.

Patrick Langer, Thomas Kaar, Max Rosenblattl, Maxwell A Xu, Winnie Chow, Martin Maritsch, Robert Jakob, Ning Wang, Juncheng Liu, Aradhana Verma, et al. Opentslm: Time-series language models for reasoning over multivariate medical text-and time-series data. *arXiv preprint arXiv:2510.02410*, 2025.

Yujie Li, Zezhi Shao, Chengqing Yu, Yisong Fu, Weijie Zhu, Yifan Du, Jilin Hu, Bin Yang, Yongjun Xu, and Fei Wang. Quals: Corpus equilibrium for universal forecasting via pattern quantization and learnability synchronization. *arXiv preprint arXiv:2609.20156*, 2026.

Zhuofeng Li, Haoxiang Zhang, Seungju Han, Sheng Liu, Jianwen Xie, Yu Zhang, Yejin Choi, James Zou, and Pan Lu. In-the-flow agentic system optimization for effective planning and tool use. In *The 5th Workshop on Mathematical Reasoning and AI at NeurIPS 2025*, 2025.

Yexiong Lin, Shanshan Ye, Yu Yao, Zhen Fang, Bo Han, and Tongliang Liu. Squidagent: Parallelize wisely, coordinate efficiently. *arXiv preprint arXiv:2610.08647*, 2026.

Haoxin Liu, Harshavardhan Kamarthi, Lingkai Kong, Zhiyuan Zhao, Chao Zhang, and B Aditya Prakash. Time-series forecasting for out-of-distribution generalization using invariant learning. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 31312–31325, 2024a.

Haoxin Liu, Zhiyuan Zhao, Jindong Wang, Harshavardhan Kamarthi, and B Aditya Prakash. Lstprompt: Large language models as zero-shot time series forecasters by long-short-term prompting. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 7832–7840, 2024b.

Penghang Liu, Elizabeth Fons, Svitlana Vyetrenko, Daniel Borrajo, Vamsi Potluru, and Manuela Veloso. Ts-agent: A time series reasoning agent with iterative statistical insight gathering. *arXiv preprint arXiv:2510.07432*, 2025.

Yong Liu, Xingjian Su, Shiyu Wang, Haoran Zhang, Haixuan Liu, Yuxuan Wang, Zhou Ye, Yang Xiang, Jianmin Wang, and Mingsheng Long. Timer-s1: A billion-scale time series foundation model with serial scaling. *arXiv preprint arXiv:2603.04791*, 2026.

Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. The m4 competition: Results, findings, conclusion and way forward. *International Journal of forecasting*, 34(4):802–808, 2018.

Martin Maritsch, Timo Stoffregen, Thomas Kaar, Behsad Riemer, Maxwell A Xu, Max Rosenblattl, Juncheng Liu, Nicolas Zumarraga, Yu Yvonne Wu, Denys Herasymuk, et al. Timenet: An extensible unified data infrastructure for next-generation temporal foundation models. *arXiv preprint arXiv:2610.04407*, 2026.

Matthew Middlehurst, Ali Ismail-Fawaz, Antoine Guillaume, Christopher Holder, David Guijo-Rubio, Guzal Bulatova, Leonidas Tsaprounis, Lukasz Mentel, Martin Walter, Patrick Schäfer, et al. aeon: a python toolkit for learning from time series. *Journal of Machine Learning Research*, 25(289):1–10, 2024.

Amin Nayebi, Sindhu Tipirneni, Chandan K Reddy, Brandon Foreman, and Vignesh Subbian. Windowshap: An efficient framework for explaining time-series classifiers based on shapley values. *Journal of biomedical informatics*, 144:104438, 2023.

Yuqi Nie, Nam H Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. In *The Eleventh International Conference on Learning Representations*, 2023.

Hao Peng, Yunjia Qi, Xiaozhi Wang, Zijun Yao, Bin Xu, Lei Hou, and Juanzi Li. Agentic reward modeling: Integrating human preferences with verifiable correctness signals for reliable reward systems. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15934–15949, 2025.

Cheng Qian, Emre Can Acikgoz, Hongru Wang, Xiusi Chen, Avirup Sil, Dilek Hakkani-Tur, Gokhan Tur, and Heng Ji. Smart: Self-aware agent for tool overuse mitigation. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 4604–4621, 2025.

Erhard Rahm, Hong Hai Do, et al. Data cleaning: Problems and current approaches. *IEEE Data Eng. Bull.*, 23 (4):3–13, 2000.

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. *Advances in neural information processing systems*, 36:68539–68551, 2023.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Lloyd S Shapley. A value for n-person games. *Contribution to the Theory of Games*, 2, 1953.

ChengAo Shen, Wenchao Yu, Fangyu Wu, Dongjin Song, Hanghang Tong, Dongsheng Luo, Wei Cheng, Haifeng Chen, and Jingchao Ni. Metacaster: Meta-harness-optimized agent for end-to-end few-shot learning of lightweight time series forecasters. *arXiv preprint arXiv:2608.23473*, 2026.

Xiaoming Shi, Shiyu Wang, Yuqi Nie, Dianqi Li, Zhou Ye, Qingsong Wen, and Ming Jin. Time-moe: Billion-scale time series foundation models with mixture of experts. In *International conference on learning representations*, volume 2025, pp. 34635–34667, 2025.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652, 2023.

Katsuto Tanaka. *Time series analysis: Nonstationary and noninvertible distribution theory*. John Wiley & Sons, 2017.

Xiaoyu Tao, Mingyue Cheng, Chuang Jiang, Tian Gao, Huanjian Zhang, and Yaguo Liu. Cast-r1: Learning tool-augmented sequential decision policies for time series forecasting. *arXiv preprint arXiv:2602.13802*, 2026.

Gemma Team. Gemma 3 technical report. *ArXiv*, abs/2503.19786, 2025. URL `https://api.semanticscholar.org/CorpusID:277313563`.

Artur Trindade. Electricityloaddiagrams20112014. *UCI Machine Learning Repository*, 10:C58C86, 2015.

Shreshth Tuli, Giuliano Casale, and Nicholas R Jennings. Tranad: Deep transformer networks for anomaly detection in multivariate time series data. *arXiv preprint arXiv:2201.07284*, 2022.

Muyao Wang, Zeke Xie, and Bo Chen. Channel-wise influence: Estimating data influence for multivariate time series. *arXiv preprint arXiv:2408.14763*, 2024.

Xi Wang and Chen Wang. Time series data cleaning: A survey. *Ieee Access*, 8:1866–1881, 2019.

Tianxin Wei, Ting-Wei Li, Zhining Liu, Xuying Ning, Ze Yang, Jiaru Zou, Zhichen Zeng, Ruizhong Qiu, Xiao Lin, Dongqi Fu, et al. Agentic reasoning for large language models. *arXiv preprint arXiv:2601.12538*, 2026.

Alexander Wettig, Aatmik Gupta, Saumya Malik, and Danqi Chen. Qurating: Selecting high-quality data for training language models. *arXiv preprint arXiv:2402.09739*, 2024.

Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in neural information processing systems*, 34: 22419–22430, 2021.

Shunyu Wu, Dan Li, Wenjie Feng, Haozheng Ye, Jian Lou, and See-Kiong Ng. Rating quality of diverse time series data by meta-learning from llm judgment. In *The Fourteenth International Conference on Learning Representations*, 2026a.

Shunyu Wu, Tianyue Li, Yixuan Leng, Jingyi Suo, Jian Lou, Dan Li, and See-Kiong Ng. Lightweight time series data valuation on time series foundation models via in-context finetuning. In *International Conference on Database Systems for Advanced Applications*, pp. 678–694. Springer, 2026b.

Xingjian Wu, Junkai Lu, Zhengyu Li, Xiangfei Qiu, Jilin Hu, Chenjuan Guo, Christian S Jensen, and Bin Yang. Timeart: Towards agentic time series reasoning via tool-augmentation. *arXiv preprint arXiv:2601.13653*, 2026c.

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.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025a.

Tiankai Yang, Junjun Liu, Michael Siu, Jiahang Wang, Zhuangzhuang Qian, Chanjuan Song, Cheng Cheng, Xiyang Hu, and Yue Zhao. Ad-agent: A multi-agent framework for end-to-end anomaly detection. In *Proceedings of the 14th International Joint Conference on Natural Language Processing and the 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics*, pp. 191–205, 2025b.

Qingren Yao, Chao-Han Huck Yang, Renhe Jiang, Yuxuan Liang, Ming Jin, and Shirui Pan. Towards neural scaling laws for time series foundation models. In *International Conference on Learning Representations*, volume 2025, pp. 83793–83820, 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 *11th International Conference on Learning Representations, ICLR 2023*, 2023.

Haobo Zhang, Yutao Zhu, Kelong Mao, Tianhao Li, and Zhicheng Dou. Recthinker: An agentic framework for tool-augmented reasoning in recommendation. *arXiv preprint arXiv:2603.09843*, 2026a.

Jingyu Zhang, Tianjian Li, William Jurayj, Hongyuan Zhan, Benjamin Van Durme, and Daniel Khashabi. Many-tier instruction hierarchy in llm agents. *arXiv preprint arXiv:2604.09443*, 2026b.

Weijia Zhang, Chenlong Yin, Hao Liu, Xiaofang Zhou, and Hui Xiong. Irregular multivariate time series forecasting: A transformable patching graph neural networks approach. In *Forty-first International Conference on Machine Learning*, 2024.

Xiaohan Zhang, Tian Gao, Mingyue Cheng, Bokai Pan, Ze Guo, Yaguo Liu, and Xiaoyu Tao. Alphacast: A human wisdom-llm intelligence co-reasoning framework for interactive time series forecasting. *arXiv preprint arXiv:2511.08947*, 2025a.

Yizi Zhang, Jingyan Shen, Xiaoxue Xiong, and Yongchan Kwon. Timeinf: Time series data contribution via influence functions. In *The Thirteenth International Conference on Learning Representations*, 2025b.

Haokun Zhao, Xiang Zhang, Jiaqi Wei, Yiwei Xu, Yuting He, Siqi Sun, and Chenyu You. Timeseriesscientist: A general-purpose ai agent for time series analysis. *arXiv preprint arXiv:2510.01538*, 2025.

Appendix

This appendix provides additional implementation details, experimental results, and analyses of TSQAgent. Section A reviews related work on time series data quality assessment and agent-based methods for time series analysis. Section B describes the construction and evaluation protocol of TSQBench. Section C presents the GRPO-based training details of the Perceiver, while Section D documents the analytical tools used by the Inspector. Section E describes the rating model architecture, training strategies, and supplementary evaluations. Section F provides experimental settings, additional results, and analyses. Section G presents qualitative case studies, followed by discussions of limitations in Section H and broader social impacts in Section I. Finally, our code is available at `https://github.com/clsr1008/TSQAgent`.

A Related Works

A.1 Time Series Data Quality Assessment

Time series data quality assessment has been widely studied in the literature. Early approaches mainly rely on statistical feature analysis and heuristic rules to detect low-quality samples, such as threshold-based anomaly detection (Clark et al., 2018), distributional checks (Tanaka, 2017), and manually designed quality indicators (Wang & Wang, 2019). While these methods are simple and interpretable, they often fail to capture complex temporal dependencies and structural distortions in real-world time series. Another line of research adapts data attribution techniques to time series for quality estimation, and can be broadly categorized into Influence Function-based (Hampel, 1974) and Shapley value-based (Shapley, 1953) approaches. For influence-based methods, TimeInf (Zhang et al., 2025b) proposes a time-aware influence function to capture temporal dependencies, while ChInf (Wang et al., 2024) extends this idea to multivariate settings via channel-level gradient approximations. LTSV (Wu et al., 2026b) further leverages in-context finetuning to approximate influence functions for efficient data valuation on time series foundation models, incorporating temporal block aggregation to preserve temporal dependencies. For Shapley-based methods, WindowSHAP (Nayebi et al., 2023) introduces window-based decomposition to enable tractable attribution over long sequences. Additionally, ShapTST (Cheng et al., 2025) integrates Shapley computation into Transformer training to jointly support forecasting and explanation. Despite their theoretical soundness, these methods are computationally expensive, as the former requires intensive Hessian and gradient computations, while the latter suffers from exponential combinatorial complexity. Recent studies have also investigated data-centric optimization for time series foundation models (TSFMs), including corpus balancing, distribution alignment, data scaling and unified data infrastructure (Li et al., 2026; Gong et al., 2026; Yao et al., 2025; Maritsch et al., 2026). These efforts highlight the importance of training data characteristics for time series modeling, complementing sample-level quality assessment and selection. More recently, motivated by the strong reasoning capabilities of large language models (LLMs), several works begin to explore LLM-based time series quality assessment (Lan et al., 2025; Cai et al., 2024). Among them, TSRating (Wu et al., 2026a) is a representative approach that defines multiple human-designed quality dimensions (e.g., trend, frequency, amplitude, and pattern consistency) and formulates pairwise comparison tasks to train a learned quality scoring model via Bradley–Terry optimization. This line of work demonstrates the potential of LLMs for semantic-level understanding of time series quality, but still heavily depends on manually specified dimensions and text-based reasoning, limiting its ability to perform fine-grained and quantitative quality assessment.

A.2 Agent-based Methods for Time Series Analysis

Recent advances have explored agentic frameworks for time series analysis, aiming to decompose complex workflows into modular reasoning and decision-making processes (Cheng et al., 2026). A line of work focuses on automating end-to-end forecasting pipelines via multi-agent collaboration (Shen et al., 2026). For example, TimeSeriesScientist (Zhao et al., 2025) introduces a four-agent system that covers data curation, model planning, forecasting, and reporting, while TimeCopilot (Garza & Rosillo, 2025) integrates multiple time series foundation models (TSFMs) with LLMs to automate feature analysis, model selection, and prediction. Similarly, AD-AGENT (Yang et al., 2025b) extends this paradigm to anomaly detection by coordinating agents for data preparation, model selection, and code generation. Another line of research emphasizes reasoning-centric agents that iteratively refine predictions or analysis through structured reasoning and interaction. AlphaCast (Zhang et al.,

2025a) formulates forecasting as an interactive co-reasoning process between human knowledge and LLMs, incorporating multi-source context and reflective optimization. TimeXL (Jiang et al., 2025) and TimeART (Wu et al., 2026c) further adopt multi-agent or multi-stage reasoning loops to improve prediction accuracy and interpretability, while TS-Agent (Liu et al., 2025) focuses on iterative evidence gathering and reasoning over raw time series signals. In addition, several works explore tool-augmented and reinforcement learning-based agent designs for time series modeling. For instance, Cast-R1 (Tao et al., 2026) reformulates forecasting as a sequential decision-making process, where an agent interacts with external tools and is optimized via reinforcement learning to iteratively refine predictions. Despite these advances, existing time series agents are primarily designed for forecasting, anomaly detection, or question answering, and focus on improving predictive performance or interpretability. In contrast, the problem of *time series quality assessment*, particularly identifying decision-relevant quality dimensions and performing grounded, quantitative comparison, remains largely underexplored in agentic settings.

B TSQBench: Benchmark Construction and Evaluation Protocol

B.1 Synthetic Time Series Template Generation

Each benchmark instance is constructed from a shared template time series, which serves as the common basis for pairwise time series. The template is designed to capture diverse yet realistic temporal structures, including trend, periodicity, and stochastic variations. Two correlated series $(\text{series}_A, \text{series}_B)$ are generated by duplicating the template, ensuring structural comparability and facilitating controlled dimension-specific degradations in subsequent steps. Specifically, each template is constructed by composing three fundamental components: a trend term, a seasonal (periodic) term, and a stochastic noise term.

To enhance diversity, multiple composition strategies are adopted, including additive combinations, multiplicative interactions between trend and seasonality, and sequential compositions that concatenate multiple segments with distinct local patterns. The trend component covers a range of forms such as linear, piecewise, exponential, and logarithmic variations, while the seasonal component includes different periodic patterns (e.g., sinusoidal, square, and mixed harmonics). The noise component introduces low-level stochastic variations with different temporal structures (e.g., independent, autocorrelated, or time-varying).

All components are sampled from continuous parameter ranges and combined with randomized scaling and shifting, resulting in a broad distribution of temporal dynamics. Importantly, the baseline noise level is intentionally kept low, as dimension-specific degradations are introduced separately in later stages. This design ensures that the template captures clean underlying structures while allowing controlled manipulation of quality factors during benchmark construction.

Table 6: Quality dimension schema used in our work.

Dimension	Description
Missing Value	Fraction of missing or unobserved values; higher ratios indicate lower quality.
Noise Level	Degree of noise or stochastic perturbations in the signal; higher levels indicate lower quality.
Rare Pattern	Anomalous observations caused by data defects (e.g., sensor glitches, corruption); more frequent or severe outliers indicate lower quality.
Trend	Clear and sustained directional movement over time; absence of stable direction (e.g., flat segments, erratic reversals) indicates low quality.
Frequency	Regular and stable oscillatory patterns over time; irregular or inconsistent cycles, such as varying periods or missing repetitions, indicate low quality.
Amplitude	Consistency and stable oscillation magnitude across cycles; varying or drifting swing strength over time indicates low quality.
Pattern Consistency	Global structural coherence over time; fragmented or unstable behavior, such as abrupt shifts or discontinuous transitions indicates low quality.

B.2 Active Quality Dimension Sampling

For each benchmark instance, a subset of dimensions defined in Table 6 is sampled to be active, where quality differences in the corresponding dimensions are introduced by degrading only one side (i.e., either series_A or series_B). We adopt a skewed distribution over the number of active dimensions in each benchmark instance to reflect real-world scenarios where quality differences are typically driven by a small number of dominant factors. The detailed distribution is shown in Table 7.

Table 7: Distribution of active dimensions per benchmark instance.

Number of Active Dimensions	Probability	Description
1	35%	Most common: single-dimension comparison
2	35%	Common: two-dimension comparison
3	20%	Moderate frequency
4	8%	Less frequent
5	2%	Rare cases

B.3 Dimension-specific Defect Injection

For each selected dimension, one of the two series is randomly assigned as the degraded side, while the other remains unchanged. We then introduce controlled, dimension-specific perturbations to the selected series to create asymmetric quality differences, while preserving the underlying shared temporal structure. The type of degradation is determined by the semantic meaning of each quality dimension, and is sampled from a predefined set of transformation operators (Xie et al., 2025). This design ensures that each dimension corresponds to a distinct and interpretable failure mode, enabling fine-grained benchmark evaluation.

For each quality dimension, we define a set of defect variants, each corresponding to a specific transformation operator (e.g., missingness injection, noise amplification, trend distortion). A variant is randomly selected during injection, and its parameters are sampled from predefined ranges. This design introduces both structural diversity and controlled controllability within each dimension. Table 8 summarizes the implemented operators for all seven dimensions. The table reports the transformation family and the parameter regimes used by the current generator.

B.4 Evaluation Tasks and Metrics

Building on the above design, which explicitly decomposes time series quality understanding into dimension identification and dimension-specific quality comparison reasoning, we evaluate models along two corresponding capabilities on a set of 1,000 synthetic samples. For dimension identification, given a pair $(\text{series}_A, \text{series}_B)$, the model is required to select a subset of relevant dimensions $\mathcal{D}_{pred}$ from the full dimension set $\mathcal{D}$, where the ground-truth set $\mathcal{D}_{gt}$ denotes the sampled active dimensions exhibiting quality differences between the two series. For quality comparison, the model is provided with the ground-truth active dimension and asked to determine which series exhibits higher quality under the specified dimension, where the ground-truth preference corresponds to the unperturbed series during defect injection. Performance is measured via Precision, Recall for dimension identification, and comparison accuracy for the quality comparison.

C Implementation Details of GRPO-based Perceiver Training

Training Target and Policy Parameterization. We fine-tune `Qwen3-4B` as the Perceiver policy using the TRL implementation of GRPO. The trained policy is parameterized with LoRA rather than full-model updates. In the default configuration, LoRA adapters are inserted into the attention projection layers `q_proj`, `k_proj`, and `v_proj`, with rank $r = 16$, scaling factor $\alpha = 32$, dropout 0.05, no bias terms, and causal-language-model task type. This keeps the trainable parameter budget small while adapting the attention subspace most directly involved in Perceiver planning.

The Perceiver is trained to output a JSON object containing a short perception summary and a list-valued `planned_dimensions` field. Only the `planned_dimensions` field is used by the GRPO

Table 8: Dimension-specific defect operators used in the synthetic benchmark.

Dimension	Operators	Semantics and parameterization
Missing Value	`random_scatter`, `burst`, `periodic`	Missing observations are inserted as isolated positions, contiguous dropout blocks, or approximately periodic gaps. The missing ratio is sampled from $[0.02, 0.08]$ for scattered and periodic masks; burst masks use 1–3 blocks with total burst ratio in $[0.02, 0.05]$. This dimension is calibrated as heavy because the missing ratio is directly observable from the input statistics.
Noise Level	`gaussian`, `heteroscedastic`, `impulsive`	Global Gaussian noise increases the whole-series residual variance. Heteroscedastic noise affects one contiguous segment, with affected length ratio $[0.10, 0.20]$ for light cases and $[0.30, 0.70]$ for heavy cases. Impulsive noise corrupts short bursts. Noise multipliers are smaller for light cases (1.2–2.0 for Gaussian/heteroscedastic; 1.5–2.0 for impulsive) and larger for heavy cases (2.5–4.0 or 3.0–5.0).
Rare Pattern	`point_outlier`, `contextual`, `level_shift`	Point outliers add one to three isolated spikes with magnitude 2.5–3.5 empirical standard deviations and are calibrated as heavy. Contextual anomalies create short V-shaped excursions of duration 2–4 and magnitude 4.0–5.0 standard deviations; these are treated as light because they are localized and shape-dependent. Temporary level shifts last 5–10 steps with magnitude 1.5–2.5 standard deviations and are calibrated as heavy.
Trend	`flatten`, `drift`, `reversal`	Trend defects modify a local structural component rather than the whole signal. Flattening replaces a segment of length ratio $[0.15, 0.40]$ by its local mean plus amplified residual noise. Drift adds a gradual slope change across a long segment spanning from the early part of the series to the later part. Reversal reverses a segment of length ratio $[0.15, 0.40]$ and level-aligns the boundary. These operators are calibrated as light because the degraded trend often preserves plausible local values.
Frequency	`competing`, `jitter`, `period_shift`	Frequency defects operate on the oscillatory component using the known base period when available. Competing-frequency defects add one weak non-harmonic sinusoid in light cases, or two to three stronger components in heavy cases. Jitter applies smooth cumulative phase noise with coefficient of variation $[0.08, 0.22]$. Period shifts modify the effective period in the last 20–50% of the series by a factor in $[0.50, 0.80]$.
Amplitude	`random_scale`, `decay`, `clip`	Amplitude defects separate the baseline from the oscillatory component before perturbation. Random scaling samples per-cycle log-normal scale factors with coefficient of variation $[0.20, 0.60]$ for light cases and $[0.80, 1.10]$ for heavy cases. Decay applies a monotone envelope that either grows or shrinks the oscillatory magnitude toward a factor in $[0.20, 0.70]$. Clipping caps oscillatory peaks and troughs at 40–80% of their original maximum magnitude.
Pattern Consistency	`variance_switching`, `structural_break`, `flat_spots`, `mean_drift`	Consistency defects disturb global structural coherence. Variance switching alternates calm and volatile windows and is calibrated as heavy. Structural breaks introduce one or more persistent level shifts in the middle portion of the series. Flat spots replace random spans, typically 5–15% of the sequence length, with a constant value. Mean drift adds a smoothed random-walk displacement to the signal level, with step scale proportional to the series standard deviation.

reward. Thus, the implemented RL objective optimizes dimension selection and output validity, not the Inspector's later tool-use decision. This distinction is important because the final TSQAgent pipeline still delegates detailed measurement and tool invocation to the Inspector, while the tailored GRPO is used to improve the Perceiver's planning precision.

Training Data. We use the synthetic time series quality benchmark as the supervision source for GRPO. Each training instance provides a pair of time series segments and a ground-truth target dimension set D_{gt}, which indicates the quality dimensions that should be selected by the Perceiver. The data construction process, defect taxonomy, label semantics, and split details are described in Appendix B.

Reward Function. The proposed TS quality-tailored GRPO reward combines a small format-compliance term with a dominant dimension-selection term. The format term encourages the model to return parseable JSON following the Perceiver schema, while the dimension term measures the precision of the predicted `planned_dimensions` against the target dimension set D_{gt}. The effective reward used by the trainer is

$$R = 0.10\, R_{\mathrm{format}} + 0.90\, R_{\mathrm{dim}}, \tag{3}$$

where R_{format} ensures valid structured output, and R_{dim} measures the alignment between predicted and ground-truth dimensions:

$$R_{\text{dim}} = \frac{|\mathcal{D}_{pred} \cap \mathcal{D}_{gt}|}{|\mathcal{D}_{pred}|}. \tag{4}$$

The 0.1/0.9 weighting reflects the role of GRPO in this project: valid structured output is necessary, but the main objective is to make the Perceiver select relevant dimensions without over-selecting. This precision-oriented design discourages forwarding redundant dimensions to the Inspector, thereby reducing downstream tool calls and irrelevant evidence.

Tailored GRPO Optimization Configuration. For each prompt, GRPO samples $K = 8$ completions from the current policy using temperature 1.0. These completions form the comparison group for relative policy optimization. With the default gradient accumulation of four steps, each optimizer update aggregates four unique prompts and 32 sampled completions per GPU. Under DDP with N GPUs, the effective update aggregates $4N$ prompts and $32N$ completions.

The training run uses one epoch, learning rate 10^{-5}, cosine learning-rate decay, and warmup ratio 0.10. The KL regularization coefficient is $\beta = 0.04$, which constrains the fine-tuned policy against the reference behavior used internally by TRL's GRPO trainer. This is important because the reward is sparse and format-sensitive: without KL pressure, the policy could overfit to short, reward-hacking completions or collapse into degenerate JSON templates. The maximum generated completion length is 300 tokens, which is sufficient for the required Perceiver JSON while limiting long reasoning traces.

Runtime and Checkpointing. The model is loaded in bf16 by default, uses the PyTorch SDPA attention implementation, and enables gradient checkpointing to reduce memory usage during GRPO sampling. The default full-training uses one prompt per GPU, eight generations per prompt, and four gradient-accumulation steps. The trainer saves LoRA checkpoints at the end of each epoch and limits the number of saved checkpoints to the number of epochs. In the full TSQAgent pipeline, the LoRA-served model can be assigned only to the Perceiver, while the Inspector and Adjudicator continue to use the base model endpoint; this isolates the effect of GRPO to the planning stage. Our training is conducted on a single NVIDIA A100 GPU.

D Implementation Details of Time Series Analytical Tools

This section describes the implementation of the 16 analytical tools used by the Inspector in TSQAgent. The tools are organized into three categories: data completeness and noise analysis, outlier and rare pattern detection, and temporal structure and pattern analysis. Each tool is accessed through a unified registry, operates on a specified time series (`series_name` $\in \{\mathrm{A}, \mathrm{B}\}$), and returns structured quantitative evidence for the Inspector's reasoning process.

All tools operate on a univariate time series $x = \{x_t\}_{t=1}^{T}$. Tools based on summary statistics ignore missing entries when computing valid statistics, while tools requiring continuous temporal structure, such as autocorrelation and trend analysis, fill missing values via linear interpolation before computation.

D.1 Data Completeness and Noise Analysis Tools

(1) Missing Ratio. The `missing_ratio` tool quantifies data completeness by counting the fraction of missing entries:

$$\texttt{missing_ratio}(x) = \frac{|\{x_t \mid x_t = \text{NaN}\}|}{T}. \tag{5}$$

It returns `missing_ratio`, where larger values indicate more severe data loss. This tool is the primary evidence source for the *Missing Value* dimension.

(2) Noise Profile. The `noise_profile` tool estimates the high-frequency component of a series by subtracting a rolling mean from the valid observations:

$$\texttt{noise_ratio}(x) = \frac{\sigma(x - \hat{x}^{(w)}_{\text{roll}})}{\sigma(x)}, \tag{6}$$

where $\hat{x}^{(w)}_{\text{roll}}$ is a moving average with window size w. If w is not specified, the implementation adapts it to the sequence length. The tool returns `noise_std`, `signal_std`, `noise_ratio`, `noise_type`, and `window_used`. The noise type is determined by the lag-1 autocorrelation of the valid series, where strongly autocorrelated residual behavior is labeled as red noise and weakly autocorrelated behavior as white noise.

(3) Volatility. The `volatility` tool measures local instability using the rolling standard deviation of first-order differences:

$$\texttt{volatility}(x) = \frac{1}{N_w} \sum_i \text{Std}\left(\Delta x_{i:i+w}\right), \quad \Delta x_t = x_t - x_{t-1}. \tag{7}$$

It returns `mean_volatility`, `max_volatility`, and `window_used`. While `noise_profile` measures residual variation after smoothing, `volatility` captures abrupt step-to-step changes, making it a complementary tool for the *Noise Level* dimension.

(4) Range Statistics. The `range_stats` tool computes local statistics over an interval $[i, j)$:

$$\texttt{range_stats}(x, i, j, s) = s(x_{i:j}), \quad s \in \{\text{mean}, \text{std}, \max, \min, \text{sum}\}. \tag{8}$$

It returns the selected statistic value together with the clipped interval and segment length. This tool is used as local supporting evidence, for example after a change point has been identified or when the Inspector needs to compare a specific region rather than the entire series.

D.2 Outlier and Rare Pattern Detection Tools

(5) Z-score Outliers. The `zscore_outlier` tool detects global point anomalies using standard deviation thresholding:

$$z_t = \frac{|x_t - \mu|}{\sigma}, \quad \mathbb{I}_{\text{outlier}}(t) = \mathbb{I}[z_t > \lambda]. \tag{9}$$

It returns `anomaly_count`, `anomaly_ratio`, `anomaly_indices`, `anomaly_values`, and `threshold_used`. This tool provides a simple first-pass detector for isolated spikes, especially when the series is approximately stationary.

(6) Outlier Density. The `outlier_density` tool estimates anomaly density using interquartile-range fences:

$$\text{IQR} = Q_3 - Q_1, \quad \ell = Q_1 - 1.5\,\text{IQR}, \quad u = Q_3 + 1.5\,\text{IQR}. \tag{10}$$

Points outside $[\ell, u]$ are counted as outliers. The tool returns `outlier_count`, `outlier_ratio`, `iqr`, `lower_fence`, and `upper_fence`. Compared with Z-score thresholding, this detector is more robust to heavy-tailed distributions because it relies on quantiles rather than mean and variance.

(7) MAD-based Residual Outliers. The `mad_residual_outlier` tool first removes local trend using a rolling mean and then applies median absolute deviation (MAD) scoring to the residuals:

$$r_t = x_t - \hat{x}^{(w)}_{\text{roll},t}, \quad m_t = \frac{0.6745\,|r_t - \text{median}(r)|}{\text{MAD}(r)}. \tag{11}$$

Points with $m_t > \lambda$ are flagged as anomalies. The tool returns `anomaly_count`, `anomaly_ratio`, `anomaly_indices`, `anomaly_values`, `mad`, `threshold_used`, and `window_used`. This robust

detector is preferred when a series has trend or seasonal drift, where global Z-scores may produce false positives.

(8) Contextual Rare Pattern. The `contextual_rare_pattern` tool identifies points that deviate from their local context. For each time index, it fits a linear model over the preceding context window and scores the prediction error:

$$\hat{x}_t = a_t t + b_t, \quad e_t = |x_t - \hat{x}_t|. \tag{12}$$

The errors are normalized using MAD, and points with unusually large local prediction errors are flagged. The tool returns `anomaly_count`, `anomaly_ratio`, `anomaly_indices`, `anomaly_values`, `threshold_used`, and `context_window_used`. This tool is useful for contextual events such as sudden drops, V-shaped deviations, or local structural breaks that may not be extreme under global statistics.

D.3 Temporal Structure and Pattern Analysis Tools

This category corresponds to tools for assessing structural richness and temporal organization, covering the *Trend*, *Frequency*, *Amplitude*, and *Pattern Consistency* dimensions.

(9) Trend Classifier. The `trend_classifier` tool estimates global trend direction and strength via linear regression:

$$x_t = \beta_0 + \beta_1 t + \epsilon_t, \quad \texttt{trend_strength} = R^2. \tag{13}$$

It returns `direction`, `slope`, `trend_strength`, `segment_clarity`, and `segment_count`. In addition to the global fit, the implementation uses change points to segment the series and computes per-segment trend clarity, which captures whether each natural segment exhibits a coherent direction even when the overall series is not globally linear.

(10) Change Point Detection. The `change_point_detector` tool detects structural breakpoints where the mean level or temporal behavior changes:

$$\texttt{change_point_detector}(x) = \{\tau_1, \tau_2, \ldots, \tau_K\}. \tag{14}$$

The implementation uses the PELT algorithm with an ℓ_2 cost when the `ruptures` package is available, and falls back to a CUSUM-style detector otherwise. It returns `change_point_count`, `change_point_indices`, and `method`. This tool supports trend, amplitude, and pattern-consistency analysis by localizing regions where the series changes character.

(11) Seasonality Detection. The `seasonality_detector` tool searches for dominant periodicity using the autocorrelation function:

$$\rho(\tau) = \frac{\mathbb{E}[(x_t - \bar{x})(x_{t+\tau} - \bar{x})]}{\mathrm{Var}(x)}, \quad \tau^\star = \arg\max_\tau \rho(\tau). \tag{15}$$

The implementation detects significant ACF peaks with prominence filtering and suppresses harmonics so that near-integer multiples of a stronger base period are not double-counted. It returns `dominant_period`, `seasonal_strength`, `top_periods`, `dominance_ratio`, and `peak_count`. This is the primary tool for the *Frequency* dimension.

(12) Autocorrelation. The `autocorr` tool computes dependency at a specified lag:

$$\texttt{autocorr}(x, \tau) = \mathrm{corr}(x_{1:T-\tau}, x_{1+\tau:T}). \tag{16}$$

It returns `lag` and `autocorrelation`. Unlike `seasonality_detector`, which searches across many lags, this tool verifies a specific lag hypothesis and is therefore used as supporting evidence for frequency and smoothness-related judgments.

(13) Pattern Consistency Indicators. The `pattern_consistency_indicators` tool computes a collection of structural coherence measures:

$$\texttt{lumpiness} = \mathrm{Var}\left(\{\mathrm{Var}(x_{W_k})\}_k\right), \tag{17}$$

$$\texttt{roughness} = \frac{1}{T-1}\sum_{t=1}^{T-1} |x_{t+1} - x_t|. \tag{18}$$

It also returns `flat_ratio`, `longest_flat_ratio`, `crossing_points`, and `crossing_rate`. These indicators capture bursty variance, stagnant intervals, frequent mean crossings, and jaggedness. They are the primary evidence source for the *Pattern Consistency* dimension.

(14) Stationarity Test. The `stationarity_test` tool provides statistical tests for stationarity. It supports the Augmented Dickey–Fuller (ADF) test and the KPSS test:

$$\texttt{stationarity_test}(x) = (\text{statistic}, p, \text{is_stationary}). \tag{19}$$

For ADF, the null hypothesis is that the series has a unit root, so $p < 0.05$ indicates stationarity. For KPSS, the null hypothesis is stationarity, so $p > 0.05$ indicates stationarity. The tool returns `test`, `statistic`, `p_value`, and `is_stationary`. It is used as supporting evidence for trend and pattern-consistency analysis.

(15) Rolling Amplitude. The `rolling_amplitude` tool computes local amplitude as the range within a sliding window:

$$\texttt{local_range}_i = \max(x_{i:i+w}) - \min(x_{i:i+w}). \tag{20}$$

It returns `mean_local_range`, `cv_local_range`, `max_local_range`, `min_local_range`, and `window`. This tool is a general-purpose amplitude measure and is especially useful when a series is not clearly oscillatory.

(16) Cycle Amplitude. The `cycle_amplitude` tool measures cycle-level amplitude consistency for oscillatory series. It first performs a gate check requiring at least two significant peaks and two significant troughs, where significance is determined by peak prominence relative to the series standard deviation. It then pairs adjacent extrema and computes peak-to-trough magnitudes:

$$a_k = |x_{p_k} - x_{q_k}|, \quad \texttt{amplitude_cv} = \frac{\sigma(\{a_k\})}{\mu(\{a_k\})}. \tag{21}$$

The tool returns `oscillatory`, `cycle_count`, `mean_amplitude`, `amplitude_cv`, `amplitude_trend`, `peak_count`, and `trough_count`. It is the primary amplitude tool when the series has clear oscillations; otherwise, the Inspector falls back to `rolling_amplitude`.

E Rating Model Design and Training Strategies

E.1 Rating Model Architecture and Training Objective

The rating model is designed to assign a scalar quality score to each time series segment. Given an input time series $\mathbf{x}$, the model outputs a scalar score reflecting its quality under predefined criteria. The architecture follows a two-stage design: representation extraction and quality mapping.

Representation Learning. We adopt a pretrained time series foundation model, MOMENT (Goswami et al., 2024), to encode input series into latent representations. MOMENT contains approximately 109M parameters and is pretrained to capture diverse temporal patterns, providing a strong feature extractor. To ensure stable and efficient training, the encoder is kept frozen during optimization.

Quality Mapping. The extracted representation is fed into a lightweight multi-layer perceptron (MLP) to produce a scalar quality score. The MLP consists of three fully connected layers with hidden dimension 256. Each layer is followed by LayerNorm and ReLU activation, and residual connections are incorporated to improve optimization stability.

The model is trained using pairwise preference data derived from agent judgments. Specifically, given a pair of time series $(\mathbf{x}_i, \mathbf{x}_j)$ with preference label $p_{i\succ j}$ indicating whether $\mathbf{x}_i$ is preferred over $\mathbf{x}_j$, the loss is defined as:

$$\mathcal{L}_\theta = \mathbb{E}_{(\mathbf{x}_i,\mathbf{x}_j,p_{i\succ j})\in\mathcal{J}}\Big[-p_{i\succ j}\log\sigma\big(s_\theta(\mathbf{x}_i)-s_\theta(\mathbf{x}_j)\big)-(1-p_{i\succ j})\log\sigma\big(s_\theta(\mathbf{x}_j)-s_\theta(\mathbf{x}_i)\big)\Big], \tag{22}$$

where $\mathcal{J}$ denotes the set of pairwise comparisons produced by the agent, and $s_\theta(\mathbf{x})$ is the predicted quality score parameterized by θ. This formulation corresponds to a standard pairwise ranking objective and is consistent with the maximum likelihood estimation.

E.2 Pairwise Comparison Sampling Details

For the main experiments in Section 4.2, we train a separate rating model for each dataset. Given a training pool containing N samples, we uniformly sample $N/2$ pairs at random without replacement for pairwise quality assessment, balancing training-set coverage and annotation efficiency. The resulting preference judgments are used as supervision for training the corresponding rating model. To reduce noisy supervision, we discard pairwise comparisons whose confidence scores estimated by the Adjudicator are below 0.5. Across all datasets, fewer than 10% of the sampled comparisons are removed by this filtering procedure. Table 9 reports the exact number of sampled pairwise comparisons for each dataset before confidence filtering.

Table 9: Number of pairwise comparisons sampled for dataset-specific rating model training before confidence filtering.

Dataset	Elec.	ExRate	Traffic	Weather	M4Y	M4M	M4D	MImg	CBF	BME	HW
Pairs	1342	1340	1343	1338	1391	1381	1379	503	435	435	500

For the cross-dataset TSFM fine-tuning experiments in Section 4.2.3, we instead train a shared rating model using the meta-learning strategy described in Appendix E.4. In this setting, we sample 500 pairwise comparisons from each dataset to construct the multi-domain training pool, allowing the rating model to leverage quality supervision across datasets and learn transferable quality representations.

E.3 Effectiveness of Pairwise Quality Assessment

We adopt pairwise comparison to elicit more reliable quality assessments from LLMs. Directly assigning numerical quality scores to individual time series requires calibrating an absolute quality scale, whereas pairwise comparison only requires relative judgments between two samples, which are generally easier and more stable to obtain. Similar pairwise preference-based paradigms have also been adopted in recent quality evaluation studies, such as QuRating (Wettig et al., 2024).

To further validate this design, we compare pairwise comparison with direct rating on the Weather dataset, where the two approaches differ in how quality judgments are elicited while following the same data selection pipeline with a 50% selection budget. We evaluate their effectiveness using three downstream forecasting models (Linear, CNN, and PatchTST) in terms of forecasting RMSE. As shown in Table 10, pairwise comparison consistently achieves lower forecasting RMSE across all three models, supporting the effectiveness of pairwise preference modeling for time series quality assessment.

Table 10: Comparison of pairwise comparison and direct rating on the Weather dataset. Results report forecasting RMSE (↓) as mean ± standard deviation over 10 random seeds. Best mean results are **bolded**.

Method	**Linear**	**CNN**	**PatchTST**
Pairwise Comparison	$\mathbf{0.593} \pm 0.016$	$\mathbf{0.723} \pm 0.015$	$\mathbf{0.450} \pm 0.012$
Direct Rating	0.607 ± 0.020	0.738 ± 0.012	0.455 ± 0.010

E.4 Cross-Dataset Quality Scoring via Meta-Learning

To enable consistent quality scoring across heterogeneous datasets, we further train the rating model using a meta-learning strategy. Unlike single-dataset training, where scores are only comparable within each dataset, meta-learning allows the model to learn a shared scoring function that generalizes across different data distributions.

In our setting, each dataset is treated as a separate task. We consider four datasets used in the main experiments (Electricity, ExchangeRate, Traffic, and Weather), and construct meta-training episodes by sampling tasks from this set. For each task, the available pairwise preference data is split into a support set and a query set. The model is first adapted to the support set with a small number of gradient updates, and then evaluated on the query set.

The meta-objective optimizes the model parameters such that the adapted model performs well on the query set across all tasks:

$$\min_{\theta} \sum_{\mathcal{T}_i} \mathcal{L}_{\mathcal{T}_i}^{\text{query}} \Big(\theta - \alpha \nabla_{\theta} \mathcal{L}_{\mathcal{T}_i}^{\text{support}}(\theta) \Big), \tag{23}$$

where θ denotes the model parameters and α is the inner-loop learning rate. This procedure encourages the model to capture dataset-invariant notions of data quality, making the resulting scores comparable across datasets.

In practice, we adopt a first-order approximation for efficiency and perform a small number of inner-loop updates. The resulting meta-trained rating model is then used to assign quality scores over the merged data pool for cross-dataset selection.

F Additional Experimental Details, Results, and Analyses

F.1 Benchmark Datasets and Baselines

Experimental Setting and Comparability. TSQAgent and TSRating (Wu et al., 2026a) both investigate LLM-based quality assessment for time series data. To facilitate direct and controlled comparisons, we adopt part of the evaluation protocol used in TSRating, particularly its benchmark datasets and data valuation baselines. Maintaining a consistent experimental setting helps isolate the contributions of the proposed agentic framework from differences in dataset selection or baseline configurations. Beyond these shared settings, TSQAgent extends the evaluation scope to examine additional capabilities, including cross-dataset data selection for time series foundation model fine-tuning (Section 4.2.3), tool-augmented reasoning (Section 4.2.2), and interpretability through intermediate dimension-level evidence (Appendix F.7).

Benchmark Datasets. Our experiments involve eleven benchmark datasets spanning different temporal characteristics and application domains, covering both forecasting and classification tasks. The forecasting benchmarks include four widely used datasets and three subsets of the M4 competition, while the classification benchmarks comprise four datasets with diverse temporal patterns.

- *Electricity* (Trindade, 2015): Hourly electricity usage measurements, expressed in kWh, are available for 321 customers over the period from 2012 to 2014. For univariate forecasting, we extract the consumption history associated with `MT_320` as the target series.
- *ExchangeRate* (Lai et al., 2018): This benchmark tracks the daily exchange rates of eight currencies from 1990 through 2016. Following the established univariate forecasting setting, we use the Singapore dollar exchange rate as the prediction target.
- *Traffic* (Lai et al., 2018): The traffic measurements consist of hourly freeway occupancy observations recorded by sensors in the San Francisco Bay Area during 2015–2016. Our experiments focus on the observations from sensor `861`.
- *Weather* (Wu et al., 2021): Meteorological observations from approximately 1,600 locations across the United States form this benchmark, covering variables such as temperature, humidity, and wind speed. We conduct the univariate experiments using `wet_bulb` as the target variable.
- *M4* (Makridakis et al., 2018): The M4 forecasting competition provides a collection of 100,000 time series sampled at different temporal frequencies, including yearly, monthly, daily, weekly, and hourly intervals. We select the *Yearly*, *Monthly*, and *Daily* subsets to evaluate data quality assessment across different forecasting frequencies.
- *MedicalImages* (Dau et al., 2019): This classification benchmark comprises more than 1,000 univariate sequences, each containing 99 observations derived from pixel intensity variations. The sequences are assigned to 10 categories representing different medical image classes.
- *CBF* (Dau et al., 2019): CBF consists of 930 synthetically generated univariate sequences, each with 128 time steps. Its three classes are characterized by distinct piecewise geometric patterns, providing a controlled setting for time series classification.
- *BME* (Dau et al., 2019): The BME benchmark contains 180 synthetic univariate sequences of length 128, organized into three classes distinguished by controlled variations in their temporal patterns.

- *Handwriting* (Bagnall et al., 2018): This multivariate classification benchmark consists of 1,000 handwriting trajectories. Each trajectory is represented by three channels over 152 time steps and belongs to one of 26 character classes.

Baselines. We evaluate TSQAgent against several established approaches to data valuation and subset selection, encompassing Shapley-based valuation, out-of-bag estimation, influence-based attribution, and random sampling. These methods provide sample-level utility estimates or selection criteria that can be used to construct training subsets.

- **DataShapley** (Ghorbani & Zou, 2019): DataShapley applies the Shapley value principle to training data valuation. It estimates the importance of an individual sample by considering its marginal contribution to predictive performance across different training subsets.
- **DataOob** (Kwon & Zou, 2023): DataOob evaluates training sample utility through out-of-bag predictions. By examining model behavior on samples excluded from particular training subsets, it estimates how individual observations contribute to predictive generalization.
- **TimeInf** (Zhang et al., 2025b): TimeInf extends influence-based data attribution to temporal data. It estimates the effects of individual training samples on model predictions while accounting for dependencies within time series.
- **TSRating** (Wu et al., 2026a): TSRating constructs a time series quality scoring function from pairwise preference supervision. It combines learned time series representations with a rating model to estimate scalar quality scores, allowing samples to be ranked according to their assessed quality.
- **Random**: Random selection serves as a reference strategy that constructs training subsets without considering sample quality. It provides a simple baseline for evaluating the benefits of quality-aware data selection.

F.2 Variability of Experimental Results Across Random Seeds

To assess the stability of the experimental results, we report the mean and standard deviation over 10 random seeds for the main data selection experiments in Table 2 and the component ablation study in Section 4.2.2. Overall, TSQAgent achieves competitive performance with relatively small variations across random seeds, further supporting the robustness of the reported improvements.

Data Selection Performance. We first report the detailed results of data selection methods with CNN as the downstream model under a 50% high-quality data budget. Tables 11 and 12 present the mean and standard deviation for forecasting and classification tasks, respectively.

Table 11: Mean $\pm$ standard deviation over 10 random seeds for the data selection experiments in Table 2, using CNN as the downstream model on forecasting datasets. Best mean results are **bolded**.

Method	Long-term (RMSE ↓)				Short-term (MAPE ↓)		
	Electricity	ExchangeRate	Traffic	Weather	M4 Yearly	M4 Monthly	M4 Daily
Random	1.592 ± 0.075	1.474 ± 0.073	0.504 ± 0.010	0.769 ± 0.033	2.332 ± 0.460	1.124 ± 0.042	1.193 ± 0.055
DataOob	1.609 ± 0.049	1.527 ± 0.042	0.552 ± 0.011	0.737 ± 0.009	2.957 ± 0.552	1.208 ± 0.067	1.346 ± 0.085
DataShapley	1.529 ± 0.060	1.598 ± 0.042	0.475 ± 0.012	0.767 ± 0.025	2.553 ± 0.548	1.117 ± 0.022	1.159 ± 0.036
TimeInf	1.530 ± 0.066	1.515 ± 0.053	0.505 ± 0.007	0.758 ± 0.022	2.289 ± 0.334	1.117 ± 0.034	1.103 ± 0.044
TSRating	1.511 ± 0.050	1.429 ± 0.036	0.428 ± 0.008	0.734 ± 0.025	1.782 ± 0.653	$\mathbf{1.075} \pm 0.056$	1.108 ± 0.051
TSQAgent	$\mathbf{1.419} \pm 0.054$	$\mathbf{1.371} \pm 0.037$	$\mathbf{0.412} \pm 0.009$	$\mathbf{0.723} \pm 0.015$	$\mathbf{1.744} \pm 0.354$	1.108 ± 0.027	$\mathbf{1.096} \pm 0.031$

Component Analysis. We further report the mean and standard deviation over 10 random seeds for three ablation studies, including GRPO training, tool-augmented quantitative comparison, and quality dimension design. As shown in Tables 13, 14, and 15, the results generally support the effectiveness of GRPO training, tool-augmented quantitative comparison, and the proposed quality dimension design, with relatively small variations across random seeds.

F.3 F1 Evaluation of Perceiver Dimension Identification

In addition to the Precision metric reported in Figure 2a, we further evaluate the Perceiver's dimension identification performance using F1 score, which jointly accounts for precision and recall. As shown

Table 12: Mean ± standard deviation over 10 random seeds for the data selection experiments in Table 2, using CNN as the downstream model on classification datasets. Best mean results are **bolded**.

Method	Classification (Accuracy ↑)			
	MedicalImages	CBF	BME	Handwriting
Random	0.554 ± 0.015	0.595 ± 0.061	0.495 ± 0.027	0.151 ± 0.009
DataOob	0.514 ± 0.014	0.663 ± 0.016	0.521 ± 0.022	0.155 ± 0.006
DataShapley	0.550 ± 0.011	0.593 ± 0.045	0.564 ± 0.028	0.159 ± 0.004
TimeInf	0.567 ± 0.006	0.563 ± 0.035	0.535 ± 0.046	0.155 ± 0.007
TSRating	0.561 ± 0.007	$\mathbf{0.679} \pm 0.018$	0.575 ± 0.018	0.159 ± 0.006
TSQAgent	$\mathbf{0.574} \pm 0.008$	0.668 ± 0.019	$\mathbf{0.580} \pm 0.021$	$\mathbf{0.162} \pm 0.005$

Table 13: Mean ± standard deviation over 10 random seeds for the Perceiver ablation study in Figure 3(a). Results report forecasting RMSE (↓).

Method	Electricity	ExchangeRate	Traffic	Weather
w/o GRPO	0.449 ± 0.018	0.243 ± 0.007	0.360 ± 0.005	0.512 ± 0.011
w/ GRPO	0.400 ± 0.017	0.218 ± 0.007	0.353 ± 0.004	0.496 ± 0.010

Table 14: Mean ± standard deviation over 10 random seeds for the tool-augmented quantitative comparison ablation study in Figure 4, using Qwen3-4B. Results report forecasting RMSE (↓).

Method	Electricity	ExchangeRate	Traffic	Weather
w/o tools	0.449 ± 0.014	0.205 ± 0.003	0.358 ± 0.003	0.515 ± 0.010
w/ tools	0.419 ± 0.018	0.213 ± 0.004	0.335 ± 0.003	0.494 ± 0.011

Table 15: Mean ± standard deviation over 10 random seeds for the quality dimension design ablation study in Figure 5. Results report forecasting RMSE (↓).

Method	Electricity	ExchangeRate	Traffic	Weather
Full	0.406 ± 0.012	0.211 ± 0.006	0.343 ± 0.013	0.499 ± 0.014
w/o Data Quality	0.426 ± 0.014	0.214 ± 0.005	0.353 ± 0.014	0.491 ± 0.018
w/o Rare Pattern	0.439 ± 0.012	0.217 ± 0.008	0.379 ± 0.014	0.509 ± 0.019
w/o Pattern Structure	0.419 ± 0.017	0.220 ± 0.005	0.382 ± 0.015	0.527 ± 0.024

in Table 16, GRPO training consistently improves F1 scores across all four backbone models, in agreement with the precision-based results. These findings further support the effectiveness of GRPO training in identifying relevant quality dimensions.

Table 16: F1 scores (↑) for Perceiver dimension identification with and without GRPO training across different backbone models.

Method	Qwen3-4B	Gemma4-E4B	Phi4-mini	Qwen3-8B
w/o GRPO	43.5%	48.2%	40.0%	48.3%
w/ GRPO	**64.6%**	**65.1%**	**62.9%**	**67.7%**

F.4 Additional Baseline Comparisons for Cross-Dataset TSFM Fine-tuning

We extend the cross-dataset TSFM fine-tuning experiments in Section 4.2.3 by including TSRating and DataShapley as additional baselines. Following the same experimental setting, we select 50% of the samples from the unified training pool for Timer-S1 fine-tuning and evaluate forecasting RMSE on each dataset. As shown in Table 17, TSQAgent achieves the lowest RMSE across all four datasets, outperforming both additional baselines and random selection. On ExchangeRate, the improvements are relatively modest across all selection methods, potentially because Timer-S1 already achieves a low RMSE of 0.228 without fine-tuning, leaving limited room for further improvement.

Table 17: Additional baseline comparisons for cross-dataset data selection in Section 4.2.3, using Timer-S1 with a 50% data selection budget. Results report forecasting RMSE (↓). Best results are **bolded**.

Selection Method	Electricity	ExchangeRate	Traffic	Weather
Untrained	0.477	0.228	0.293	0.510
Random	0.448	0.225	0.290	0.492
TSRating	0.434	0.224	0.286	0.487
DataShapley	0.437	0.225	0.289	0.485
TSQAgent	**0.425**	**0.221**	**0.283**	**0.484**

F.5 Generalization Scaling with Meta-Training Diversity

The cross-dataset selection experiment in Section 4.2.3 meta-trains the rating model on four forecasting datasets (Electricity, ExchangeRate, Traffic, and Weather), primarily with evaluation conducted within the same forecasting task pool. We further investigate whether the learned quality representations can transfer to unseen domains and, more importantly, how such generalization scales with the diversity of the meta-training pool.

Cross-Domain Generalization. We first evaluate the original meta rater trained on the four forecasting datasets (***Meta Rater, 4 datasets***) on three unseen classification datasets: MedicalImages, CBF, and BME. For each unseen dataset, we perform 5-shot adaptation and use the adapted rater to select the top 50% of training samples for a CNN classifier. As shown in Table 18, the 4-dataset meta rater shows a moderate degradation compared with the ***dataset-specific rater*** trained directly on each target dataset. However, it performs better than most baseline methods, suggesting it has primary generalization ability.

Scaling Generalization through a More Diverse Task Pool. We next investigate whether increasing the diversity of the meta-training task pool improves cross-domain generalization. We expand the meta-training pool using the Time-300B corpus (Shi et al., 2025), selecting 23 subsets spanning 9 domains with no overlap with the three evaluation datasets. For each subset, we collect 500 pairwise quality judgments using TSQAgent, resulting in 11,500 pairwise comparisons for meta-training. We denote the resulting model as ***Meta Rater, 23 datasets***.

As shown in Table 18, this scaled meta rater (***Meta Rater, 23 datasets***) achieves comparable performance to the Dataset-specific Rater and consistently outperforms Meta Rater (4 datasets) on unseen classification datasets. These results indicate a positive generalization scaling trend: increasing task diversity in meta-training can improve the generalization capability of the rating model and enable more transferable quality assessment capabilities.

Table 18: Cross-domain generalization of meta-trained rating models on three unseen classification datasets. Results report downstream classification accuracy (mean ± std.) after using each method to select the top 50% of training samples for CNN training.

Method	MedicalImages	CBF	BME
Random	0.554 ± 0.015	0.595 ± 0.061	0.495 ± 0.027
DataOob	0.514 ± 0.014	0.663 ± 0.016	0.521 ± 0.022
DataShapley	0.550 ± 0.011	0.593 ± 0.045	0.564 ± 0.028
TimeInf	0.567 ± 0.006	0.563 ± 0.035	0.535 ± 0.046
TSRating	0.561 ± 0.007	0.679 ± 0.018	0.575 ± 0.018
TSQAgent (Dataset-specific)	0.574 ± 0.008	0.668 ± 0.019	0.580 ± 0.021
TSQAgent (Meta Rater, 4 datasets)	0.561 ± 0.008	0.659 ± 0.015	0.568 ± 0.025
TSQAgent (Meta Rater, 23 datasets)	0.571 ± 0.009	0.673 ± 0.021	0.581 ± 0.018

Generalization and Computational Scalability. As discussed in Section 4.2.4 and Table 4, the computational cost of TSQAgent is primarily dominated by one-time pairwise judgment annotation, while the trained meta rater can be reused with lightweight adaptation and inference. Expanding the meta-training pool from 4 datasets to 23 subsets (a 5.75× increase) requires approximately 8 hours to collect 11,500 pairwise judgments. Despite the increased annotation cost, the expanded meta rater

achieves improved cross-domain generalization (Table 18) without increasing per-sample inference complexity. These results demonstrate the potential of scaling meta-training diversity to improve transferability while maintaining efficient deployment.

F.6 Generalization from Synthetic Supervision to Real-World Anomalies

In TSQAgent, the Perceiver is trained with GRPO using synthetic perturbations from TSQBench, where the active quality dimensions are explicitly controlled during data generation detailed in Appendix B. To examine whether the quality reasoning learned from such synthetic supervision can generalize beyond synthetic perturbations, we further evaluate both the Perceiver and the complete TSQAgent framework on three real-world anomaly detection datasets: SWaT (Goh et al., 2016), WADI (Ahmed et al., 2017), and MSL (Hundman et al., 2018). SWaT and WADI are real-world industrial sensor datasets, where anomalies reflect correlated multi-sensor changes and complex operational failures, while MSL is a spacecraft telemetry dataset containing diverse anomaly patterns arising from complex system dynamics.

Perceiver Generalization to Real-World Quality Patterns. We first evaluate whether the dimension selection capability learned by the GRPO-trained Perceiver transfers to real-world anomaly patterns. Since ground-truth dimension-level annotations are unavailable for these datasets, we use a specialized time series toolkit (Middlehurst et al., 2024) to construct weak dimension-level annotations. Specifically, the toolkit establishes normal behavior references from normal windows and identifies relevant dimensions by comparing dimension-wise deviation scores between normal–anomalous pairs, with thresholds estimated from normal–normal comparisons.

As shown in Table 19, the GRPO-trained Perceiver consistently improves dimension identification over its non-GRPO counterpart across all three datasets. Precision improves from 30.4% to 48.7% on SWaT, from 34.8% to 53.6% on WADI, and from 35.0% to 50.3% on MSL. Similar improvements are observed in F1, suggesting that the quality dimension selection capability learned through synthetic GRPO supervision can partially transfer to real-world anomaly patterns.

Table 19: Dimension identification performance of the Perceiver on real-world anomaly datasets using weak dimension-level annotations. Best results are **bolded**.

	SWaT	**WADI**	**MSL**
Precision (w/o GRPO)	30.4%	34.8%	35.0%
Precision (w/ GRPO)	**48.7%**	**53.6%**	**50.3%**
F1 (w/o GRPO)	34.2%	39.0%	39.1%
F1 (w/ GRPO)	**53.1%**	**57.6%**	**54.7%**

We also note that the Perceiver performs worse on these real-world anomaly datasets than on our TSQBench (Figure 2a), suggesting that the current synthetic training data may not fully cover the complex and correlated quality patterns encountered in real-world scenarios. Nevertheless, the consistent improvement over the non-GRPO variant indicates that the dimension selection capability learned from synthetic perturbations remains transferable to real-world data.

Generalization of TSQAgent to Real-World Anomalies. We further compare TSQAgent with both time series quality assessment methods (DataOob (Kwon & Zou, 2023) and TimeInf (Zhang et al., 2025b)) and specialized anomaly detection methods (TranAD (Tuli et al., 2022) and USAD (Audibert et al., 2020)) on SWaT, WADI, and MSL. DataOob and TimeInf assess sample quality from predictive utility and sample influence perspectives, respectively, while TranAD (a Transformer-based model) and USAD (an adversarial autoencoder-based model) identify anomalies by learning normal patterns and detecting deviations. TSQAgent instead learns a rater from agent-generated quality preferences over unlabeled sample pairs and identifies anomalies as samples with low quality scores.

For a fair comparison, all methods use only unlabeled sequences for training, scoring, and threshold selection, while test labels are accessed only for final F1 computation. As shown in Table 20, TSQAgent achieves the highest F1 scores across all three datasets. The improvements are particularly notable on MSL and SWaT, where TSQAgent surpasses the strongest competing methods by 13.2 and 12.4 F1 points, respectively, while also achieving the best performance on WADI. These results provide further evidence that the quality rationale learned by TSQAgent from synthetic perturbations

can extend beyond isolated synthetic defects to real-world data-quality failures with complex anomaly patterns.

Table 20: Anomaly detection performance (F1 scores) on three real-world datasets. Best results are **bolded**.

Method	MSL	SWaT	WADI
DataOob	0.575	0.686	0.671
TimeInf	0.654	0.473	0.735
TranAD	0.518	0.323	0.686
USAD	0.535	0.304	0.686
TSQAgent	**0.786**	**0.810**	**0.752**

The strong performance of the complete TSQAgent framework with the Perceiver-level unit test suggests that real-world quality assessment benefits from the collaborative design of the three agents. In particular, the Perceiver selects potentially relevant quality dimensions, the Inspector provides quantitative dimension-level evidence through analytical tools, and the Adjudicator integrates these intermediate results to make the final quality judgment.

Discussion and Future Directions. Due to the complexity of temporal patterns, obtaining reliable fine-grained quality dimension annotations for real-world time series requires substantial expert effort and annotation costs. In contrast, synthetic perturbations provide controllable and accurate ground-truth supervision at a relatively low cost, enabling a rigorous investigation of whether the Perceiver can identify relevant quality dimensions. Nevertheless, synthetic perturbations alone may not fully capture the diverse and complex quality patterns encountered in real-world scenarios. Incorporating real-world data into Perceiver training therefore represents a promising direction for improving its generalization capability. Importantly, the proposed training pipeline is not inherently limited to synthetic data and can naturally incorporate real-world quality patterns once reliable dimension-level annotations become available.

F.7 Dimension-level Interpretability and Task-dependent Quality Assessment

Beyond overall pairwise quality assessment, TSQAgent preserves intermediate dimension-level information that supports interpretable quality judgments. We further investigate how the significance of different quality dimensions varies across downstream tasks through two case studies on anomaly detection and forecasting.

Dimension-level Interpretability. TSQAgent provides interpretable dimension-level evidence throughout its three-agent reasoning process. Specifically, the Perceiver identifies and ranks relevant quality dimensions for each pairwise comparison, retaining the selected dimensions for subsequent analysis. The Inspector then evaluates each selected dimension using analytical tools and generates dimension-specific evidence, including statistical measurements, detected temporal patterns, and comparative assessments. Finally, the Adjudicator integrates these intermediate results to produce the final pairwise quality judgment. By preserving the dimension selections, rankings, and supporting evidence, TSQAgent enables further inspection and interpretation of its quality assessment decisions.

Task-dependent Dimension Significance. To investigate whether TSQAgent captures task-dependent quality characteristics, we analyze the significance distributions of different quality dimensions on two representative datasets: MSL (Hundman et al., 2018) for anomaly detection and Electricity (Trindade, 2015) for forecasting.

On the MSL spacecraft telemetry dataset, Rare Pattern emerges as the dominant quality dimension in comparisons involving anomalous samples. As shown in Table 21, Rare Pattern accounts for 84.95% of the dimension significance in normal–anomalous comparisons and 72.92% in anomalous–anomalous comparisons, compared with only 10.84% in normal–normal comparisons. This observation is consistent with the characteristics of anomaly detection, where unusual temporal behaviors are particularly informative. Moreover, when evaluating normal–anomalous pairs under the Rare Pattern dimension, TSQAgent correctly identifies normal samples as higher-quality in 96.20% of comparisons, demonstrating the utility of its dimension-level quality judgments.

Table 21: Distribution of quality dimension significance across different comparison types on the MSL anomaly detection dataset. The last column reports pairwise quality assessment accuracy under the Rare Pattern dimension for normal–anomalous comparisons.

Comparison	Trend	Amp.	Pattern	Freq.	Noise	Missing	Rare	Rare Acc.
Normal–Normal	14.61%	23.43%	20.91%	12.85%	14.36%	3.00%	10.84%	–
Normal–Anomaly	1.08%	4.30%	3.23%	0.00%	5.38%	1.08%	**84.95%**	**96.20%**
Anomaly–Anomaly	2.08%	8.33%	9.38%	0.00%	4.17%	3.13%	**72.92%**	–

In contrast, the Electricity forecasting dataset exhibits a different dimension significance distribution. As shown in Table 22, Trend and Amplitude are the two most significant dimensions, accounting for 34.66% and 30.16%, respectively, or 64.82% combined. Meanwhile, Rare Pattern contributes only 0.35%. This distribution aligns with the characteristics of electricity forecasting, where changes in temporal trends and signal amplitudes are particularly relevant to prediction.

Table 22: Distribution of quality dimension significance on the Electricity forecasting dataset.

Dimension	Trend	Amp.	Pattern	Freq.	Noise	Missing	Rare
Significance	**34.66%**	**30.16%**	13.17%	11.44%	8.67%	1.56%	0.35%

These case studies demonstrate that TSQAgent can identify different dominant quality dimensions across downstream tasks while preserving interpretable evidence for dimension-level assessment.

Discussion on Quality Aggregation. In the current evaluation, we primarily aggregate dimension-level evidence into overall quality judgments to enable direct comparisons with existing data valuation methods. Nevertheless, TSQAgent retains the intermediate dimension selections, rankings, and analytical evidence, allowing its quality assessments to be interpreted at the dimension level. These capabilities also provide a basis for future task-dependent quality assessment, where different quality dimensions may be emphasized according to specific application requirements.

G Qualitative Case Studies

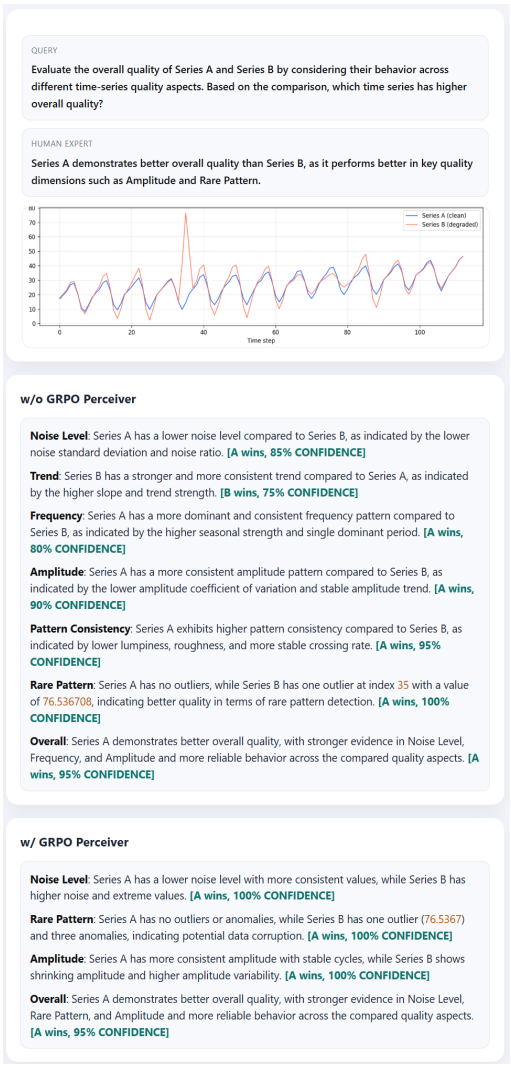
We further present two qualitative case studies to provide an intuitive understanding of how GRPO-based training and tool usage improve the efficiency and reliability of TSQAgent. The first study focuses on the effect of GRPO on dimension selection behavior, while the second examines the impact of tool-augmented quantitative comparison.

G.1 Case Study I: Effect of GRPO-based Dimension Selection.

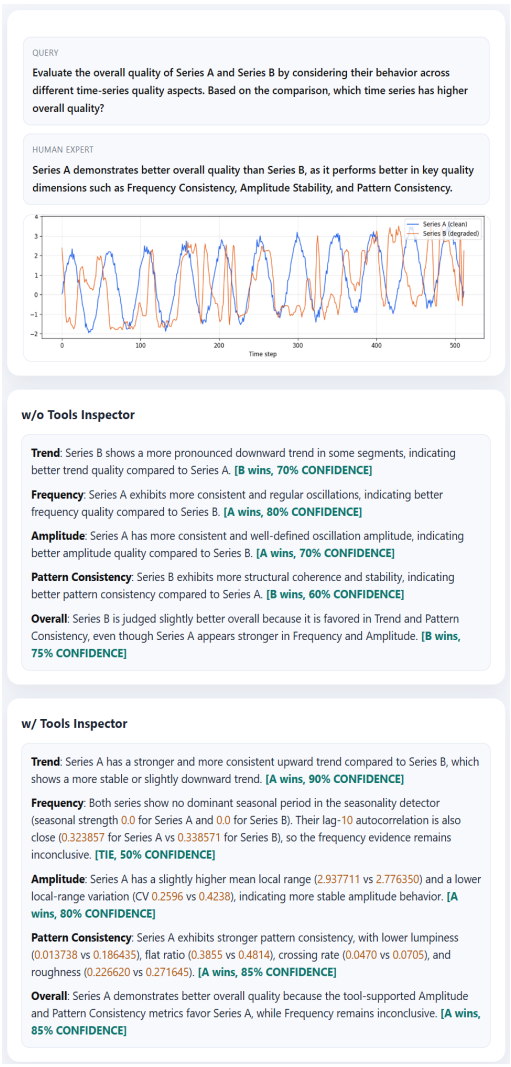
As shown in Figure 6a, without GRPO Perceiver, the Agent tends to over-select multiple quality dimensions and produces redundant or less informative analyses. This leads to inefficient reasoning and less precise judgments. In contrast, with GRPO Perceiver, the Agent focuses on a smaller set of truly relevant quality dimensions and provides more targeted and accurate assessments. This case demonstrates that GRPO Perceiver improves both the precision and efficiency of quality reasoning by reducing unnecessary dimensions and emphasizing the most discriminative factors.

G.2 Case Study II: With vs. Without Tool Augmentation.

As shown in Figure 6b, when tool usage is disabled, the Inspector mainly relies on subjective reasoning, leading to inconsistent evaluations across quality aspects. It overemphasizes trend and pattern consistency, resulting in an incorrect overall judgment favoring Series B. When tool support is enabled, the Inspector incorporates quantitative measurements from external tools, leading to more grounded and consistent assessments. As a result, it correctly identifies Series A as having higher overall quality. This case shows that tool-based evidence improves the reliability and precision of the Inspector by reducing subjective bias.



(a) GRPO-based Dimension Selection



(b) Tool-Augmented Quality Comparison

Figure 6: Case studies on tool augmentation and GRPO-based dimension selection. Each subplot shows results on a representative time series quality comparison case, including the query, human expert judgment, and agent reasoning under different configurations. Subplot (a) compares the Agent with and without the GRPO Perceiver, demonstrating that GRPO-based dimension selection reduces redundant quality dimensions and improves the focus of quality reasoning. Subplot (b) compares the Inspector with and without tool support, demonstrating that tool-augmented quantitative measurements provide more grounded evidence and improve the reliability of the final quality judgment.

G.3 Extended Case Studies

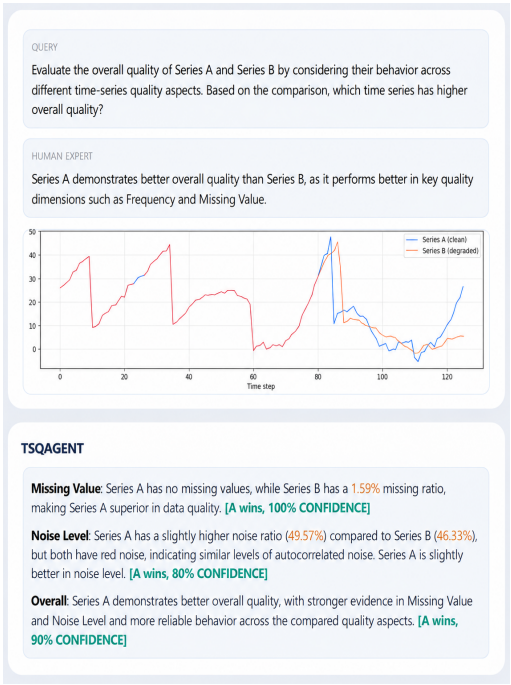
We provide extended qualitative case studies, as shown in Figure 7. These cases further demonstrate the effectiveness of our framework in identifying relevant quality dimensions and producing consistent judgments under diverse time series degradation scenarios.



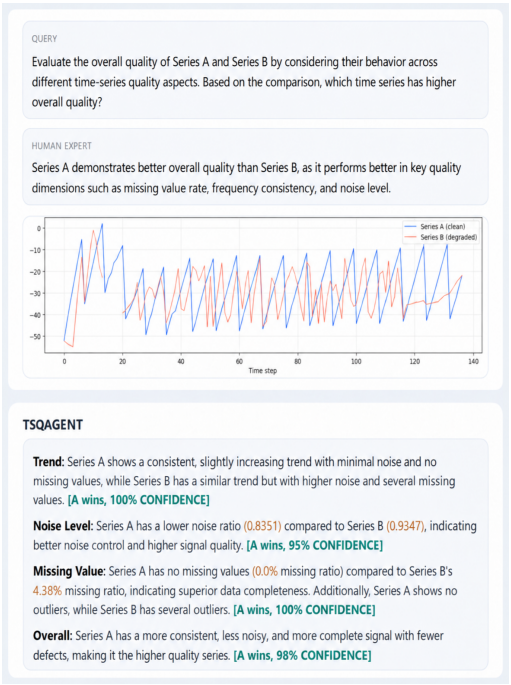
(a) Case involving pattern consistency and noise level



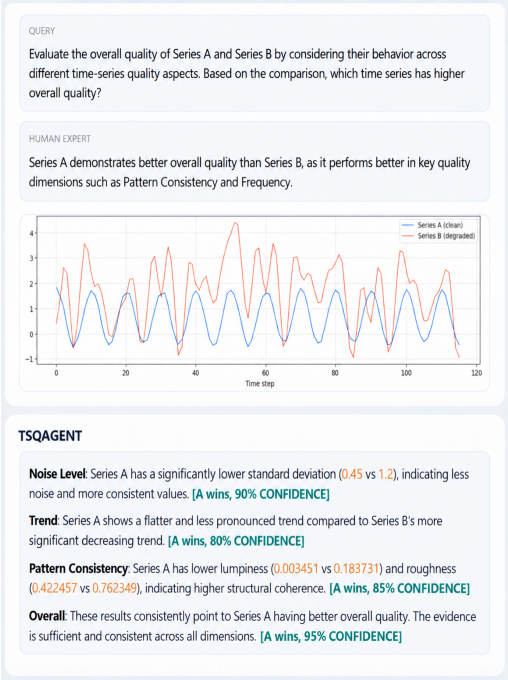
(b) Case involving missing value and rare pattern



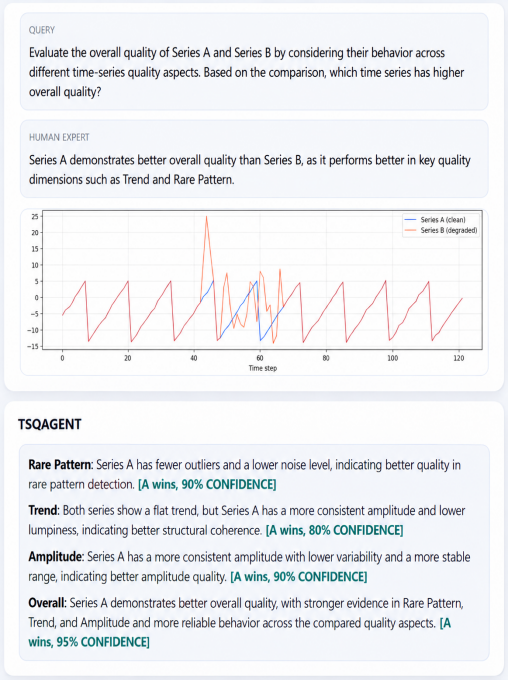
(c) Case involving frequency and missing value



(d) Case involving missing value, pattern consistency and noise level



(e) Case involving pattern consistency and frequency



(f) Case involving trend and rare pattern

Figure 7: Extended qualitative case studies under six representative quality comparison scenarios. Each subfigure illustrates the reasoning process and final judgment produced by TSQAgent.

H Limitations

Despite the effectiveness of TSQAgent, several limitations remain that warrant further investigation. First, TSQBench relies on controlled synthetic defect injection to construct quality differences between time series. Although this design enables systematic supervision and evaluation across predefined quality dimensions, synthetic perturbations may not fully reflect the complexity of real-world data quality issues, where multiple defects can interact and their manifestations may vary across domains. Consequently, the Perceiver's learned dimension selection policy may be influenced by the distributions and degradation patterns represented in TSQBench, potentially limiting its generalization to unseen or domain-specific quality issues. Future work could incorporate more diverse real-world quality annotations and degradation patterns to reduce this dependence on synthetic supervision.

Second, TSQAgent relies on a predefined set of seven quality dimensions, which may not fully capture the diverse quality characteristics encountered in different application domains. In particular, certain domain-specific quality issues may fall outside the current dimension set, potentially limiting the framework's applicability to more specialized scenarios. Adapting the framework to such scenarios may therefore require manually identifying and incorporating additional quality dimensions. Future work could explore more flexible and extensible quality dimension definitions to accommodate diverse data characteristics and application requirements.

Third, the tool-augmented Inspector introduces additional inference costs through iterative LLM reasoning and analytical tool invocation. These costs can increase with the number of selected quality dimensions, tool calls, and reflection steps, particularly when assessing large collections of time series. Although bounded reasoning steps and caching help control computational overhead, applying the full agentic pipeline to every candidate pair may still be expensive in large-scale data selection scenarios. Future work could explore more efficient tool selection, early stopping, and cost-aware agent coordination (Lin et al., 2026) to reduce inference overhead.

Finally, the scalability of TSQAgent to substantially larger data pools remains to be systematically evaluated. Although the rating model supports sample-level quality scoring and ranking, obtaining sufficient pairwise judgments for large-scale data selection may introduce additional computational

costs and sampling challenges. Future work could investigate more efficient comparison sampling and ranking strategies, while further evaluating the framework across a broader range of downstream tasks and application domains.

I Broader Social Impacts

This work focuses on time series quality assessment and data selection using large language models and agent-based reasoning. The proposed framework may have positive societal impacts by improving the efficiency and reliability of time series modeling pipelines, which are widely used in real-world applications such as energy systems, finance, healthcare, and environmental monitoring. By enabling more effective identification of high-quality data, the method can contribute to more accurate forecasting and decision-making in these domains.

At the same time, potential negative impacts should also be considered. Since the proposed method involves automated evaluation and selection of training data, improper use may amplify biases present in the data or lead to suboptimal model behavior if deployed without proper validation. In high-stakes applications, reliance on automated quality assessment without human oversight may introduce risks, especially when data distributions are shifted or noisy. Therefore, we recommend that the system be used with human-in-the-loop verification in sensitive scenarios.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The main claims are clearly stated in the Abstract and Introduction, and are supported by the proposed method in Section 3, and experimental results in Section 4.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The paper includes a dedicated Limitations discussion in Section H.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

Justification: The paper does not include formal theoretical results or proofs; it focuses on empirical evaluation of LLM-based time series quality assessment and agent design.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: All implementation details of TSQBench, GRPO-based training, tool design, and experimental configurations are provided in the main paper and Appendix B, Appendix C, and Appendix D, enabling reproduction of the reported results.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: The code and data are publicly available, with an anonymous link provided in the abstract.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Training and evaluation details are provided in Section 4, Appendix C, and Appendix E.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: All main experimental results are averaged over 10 independent runs with different random seeds to account for stochasticity in model training and evaluation. The reported values represent the mean performance across runs, as described in Section 4.2.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The paper reports the computational settings, including GPU types, runtime, and total execution cost. In addition, we provide statistics on total runtime and token consumption for LLM-based evaluations, as described in Section 4.2.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conforms to the NeurIPS Code of Ethics. All datasets used are either publicly available benchmarks or synthetically generated without personal or sensitive information, and all experiments are conducted on anonymized and non-sensitive data following standard research practices.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

Justification: The potential social impacts are discussed in Appendix I.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: The paper does not release high-risk models or datasets that require additional safeguards. The proposed method is an LLM-based reasoning framework built on existing models and publicly available benchmarks, without introducing new generative models or sensitive data.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: All datasets used in this work are publicly available and properly cited in the main paper. We follow their original licenses and usage terms. All pretrained models (including open-source and API-based LLMs) are used in accordance with their respective usage policies. No proprietary or restricted datasets are used.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The paper introduces new assets including the TSQBench benchmark, the TSQAgent framework, and the associated tool set. These components are fully described in the main paper and detailed in the appendix, with implementation details and construction procedures provided alongside the released code package for reproducibility.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing experiments or any research with human subjects. All evaluations are conducted on synthetic benchmarks and publicly available real-world time series datasets.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: The paper does not involve any human subject research or crowdsourcing experiments; therefore, IRB approval or equivalent ethical review is not applicable.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: Large language models are a core component of the proposed framework, used for perception, reasoning, and structured comparison in TSQAgent. We explicitly describe their role in all agent modules in Section 3, including how LLMs are combined with GRPO training and tool-augmented reasoning.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.