TIMESERIESEXAMAGENT: CREATING TIME SERIES REASONING BENCHMARKS AT SCALE

Anonymous authors
Paper under double-blind review

ABSTRACT

Large Language Models (LLMs) have shown promising performance in time series modeling tasks, but do they truly understand time series data? While multiple benchmarks have been proposed to answer this fundamental question, most are manually curated and focus on narrow domains or specific skill sets. To address this limitation, we propose scalable methods for creating comprehensive time series reasoning benchmarks that combine the flexibility of templates with the creativity of LLM agents. We first develop `TimeSeriesExam`, a multiple-choice benchmark using synthetic time series to evaluate LLMs across five core reasoning categories: *pattern recognition*, *noise understanding*, *similarity analysis*, *anomaly detection*, and *causality*. We then scale our approach by automatically generating benchmarks from real-world datasets spanning healthcare, finance and weather domains. Through multi-dimensional quality evaluation, we demonstrate that our automatically generated benchmarks achieve diversity comparable to manually curated alternatives. However, our experiments reveal that LLM performance remains limited in both abstract time series reasoning and domain-specific applications, highlighting ongoing challenges in enabling effective time series understanding in these models.

1 INTRODUCTION

Recent studies have successfully applied Large Language Models (LLMs) to time series analysis tasks including forecasting, anomaly detection, and classification (1; 12; 7; 17; 48; 49). However, these promising results raise a fundamental question: do LLMs possess genuine reasoning capabilities about the abstract concepts underlying time series data? Can they recognize trends, distinguish signal from noise, or understand causal relationships without relying on domain-specific shortcuts? Existing benchmarks designed to evaluate such capabilities face significant limitations– they are manually curated, expensive to extend, and typically focus on narrow domains or specific skills (42; 31). This creates a practical barrier for researchers and practitioners who need comprehensive evaluation tools but lack the resources to construct domain-specific benchmarks for their datasets.

To address this gap, we begin with a simple proof of concept, and introduce `TimeSeriesExam`, a controlled benchmark which uses *synthetic* time series to evaluate LLM reasoning across five core categories: *pattern recognition, noise understanding, similarity analysis, anomaly detection*, and *causality*. Initial results reveal two key insights: first, templated generation provides a viable mechanism for creating diverse and systematic evaluation questions; second, LLMs continue to struggle with abstract time series reasoning, even in these controlled settings.

Building on these insights, we tackle a broader practical challenge: how can we create benchmarks that reflect the domain-specific reasoning required in real applications, such as diagnosing arrhythmias from ECG signals or evaluating volatility regimes in financial markets? The key obstacle is that domain experts lack the time to manually construct comprehensive benchmarks, making expert-driven approaches impractical at scale. Inspired by recent advances in agent-based benchmark construction (5; 14), we address this challenge by combining our controlled synthetic benchmark with an extensible, agentic framework for automated domain-specific evaluation.

Experiments on multiple datasets spanning diverse domains reveal that (1) LLM performance varies substantially across domains, and (2) even state-of-the-art models struggle with complex reasoning tasks requiring integration of domain expertise and time series understanding.

Our contributions are as follows:

- **Foundational benchmark.** We introduce `TimeSeriesExam`, a controlled evaluation framework that systematically assesses whether LLMs understand core time series concepts. This templated approach provides valuable insights into LLMs' reasoning capabilities while enabling scalable question generation, though it remains limited to domain-agnostic skills on synthetic data.
- **Scalable framework.** Building on this foundation, we propose `TimeSeriesExamAgent`, which combines the systematic nature of templates with the adaptability of LLM agents. Given any domain-specific dataset, `TimeSeriesExamAgent` automatically generates customized time series reasoning benchmarks at scale, integrating multi-perspective verification and optional human-in-the-loop refinement to ensure question quality and diversity.
- **Multi-dimensional evaluation.** We validate our approach across five datasets spanning four domains and demonstrate its effectiveness for domain-specific fine-tuning. Our automatically generated questions achieve diversity comparable to human-curated benchmarks.

2 Related Work

Synthetic Time Series Generation The generation of synthetic time series with controlled behaviors, such as trends and cyclic patterns, is fundamental for constructing accurate reasoning benchmarks like `TimeSeriesExam`. A common approach involves sampling from diverse random processes (16), such as Autoregressive Processes, which offer variability but lack control over specific patterns like cyclic behavior. To address this, (46) proposed a decomposition-based method, generating desired patterns by incorporating cyclic components into an additive model on top of random processes. We build upon both works by having a more diverse set of random processes and patterns, incorporating not only additive composition methods but also multiplicative and other forms of composition.

Domain Specific Time Series Reasoning Benchmarks The task of creating domain-specific time series reasoning benchmarks is challenging. Current Domain-specific benchmarks usually have limited scope and poor extensibility, since their curation often relies on templates or expert annotation. For instance, ECG-QA (31) and ECG-Expert-QA (42) focus on ECG interpretation, while EngineMT-QA (43) targets industrial settings. Automatic benchmark generation offers a scalable alternative but raises concerns about quality and diversity of generated questions. Without extensive verification, LLM-generated questions often require heavy manual curation (19; 23), which is both difficult and time-consuming, undermining the main advantage of automation.

Agents for benchmark creation An AI agent is an autonomous system that can observe its environment, reason about possible actions, and act toward achieving a goal. Recent work has shown success in using agents for automatic benchmark creation. Most solutions adopt a multi-agent pipeline with planning, generation, validation, and evaluation modules (5). For instance, (40)

Title	Multi-Domain	Curation	# Samples	Skill type		
		Fully Automatic		P	R	PS
Time-MQA (19)	✓	✓	200,000	✓	✓	✗
Time-MMD (23)	✓	✗	17,113	✓	✗	✗
MT-Bench (8)	✓	✗	22,000	✓	✓	✗
ECG-QA (31)	✗	✗	414,348	✓	✓	✓
Context-is-key (45)	✓	✗	71	✓	✓	✗
TimeSeriesExamAgent (ours)	✓	✓	600+	✓	✓	✓

Table 1: Overview of time-series and multimodal datasets with curation and skill types (P—Prediction, R—Reasoning, PS—Practical skills). Prediction refers to supervised tasks such as forecasting or classification. Reasoning involves higher-level interpretation of time-series signals (e.g., trend recognition). Practical skills extend reasoning into domain-specific contexts (e.g., classifying volatility regimes in finance). Note that the CiK benchmark contains 71 tasks, with the number of samples treated as a configurable hyperparameter.

integrates exploratory evaluation using reinforcement learning, while (5) takes a natural language task description as input. However, most of these approaches are not tailored to time series and struggle to generate questions conditioned on numeric data. One recent solution does incorporate time series but is limited to single-step design and lacks extensive verification (27).

LLM-as-a-Judge Evaluation LLM-as-a-judge refers to using LLMs for automatic evaluation. This approach enables the research community to scale evaluations and to assess problems that are inherently difficult to capture with conventional metrics. However, LLM-as-a-judge also introduces challenges related to bias. We address this in two ways. First, we adopt methods such as G-Eval (24), which provide probabilistic results and thus improve both reliability and interpretability. Second, we account for intra-model bias, which is the tendency of a single LLM to exhibit systematic preferences, by drawing on the idea of panel-based evaluation (37). Specifically, we aggregate judgments from multiple LLMs, which reduces the influence of any individual model's bias and yields more stable scores.

3 TIMESERIESEXAMAGENT

3.1 PROOF-OF-CONCEPT BENCHMARK: TIMESERIESEXAM

To begin, we investigate LLMs' understanding of fundamental time series concepts in a controlled experimental setting by introducing a manually curated, configurable benchmark, `TimeSeriesExam`.

Composition. The `TimeSeriesExam` systematically assesses whether LLMs **understand** basic time series patterns such as trends and seasonality (*pattern recognition*), the concept of noise and other time series concepts in the presence of noise (*noise understanding*). It also evaluates LLMs on three different **reasoning** tasks: identifying abrupt deviation from "normal" behavior (11) (*anomaly detection*), comparing and contrasting statistical properties of 2 time series (*comparative reasoning*), reasoning about causality, specifically Granger Causality (13) (*causality*). As shown in Table 2, each category is further divided into sub-categories that represent more specific concepts within the broader category.

Question Templates. The `TimeSeriesExam` comprises over 100 unique templates, carefully curated in collaboration with time series experts and cross-verified for accuracy, that can be used to generate any number of random questions. Each template (Fig. 1)(*Right*) evaluates a specific (sub-)category (e.g., *pattern recognition*), and comprises of a question (e.g., `"Is this time series stationary?"`), a list of options (e.g., `"(A) Yes, (B) No"`), and an example question and answer pair for in-context learning. Each template comes with a *hint* which breaks down complex

Category	Subcategory	Example question
Pattern Recognition	Trend	What is the most likely linear trend coefficient of the given time series?
	Cyclic	The given time series has sine wave pattern. How does its amplitude change from the beginning to the end?
	Stationarity	Is the given time series likely to be stationary after removing the cycle component?
	Regime Switching	Based on the given time series, how many different regimes are there?
	Statistical properties	Is the mean stable over time in the given time series?
	Random processes	Does the following time series exhibit a mean reversion property?
Noise Understanding	White Noise	Is the given time series a white noise process?
	Random Walk	Is the given time series likely to be a random walk process?
	Signal / Noise Ratio	You are given two time series with the same underlying pattern but different noise level. Which time series has higher magnitude of noise?
Anomaly Detection		The following time series has two types of anomalies appearing at different time points. What are the likely types of these anomalies?
Comparative Analysis	Shape	Despite the noise, do the given two time series have similar patterns?
	Distributional	You are given two time series which are generated using a random walk. Are they likely to have the same variance?
Causality Analysis	Granger Causality	Is there Granger causality between the two time series?

Table 2: Example template questions for different reasoning tasks. Each subcategory covers a specific aspect of time series understanding, guiding the model to reason about comparative, anomalies, and causal relationships.

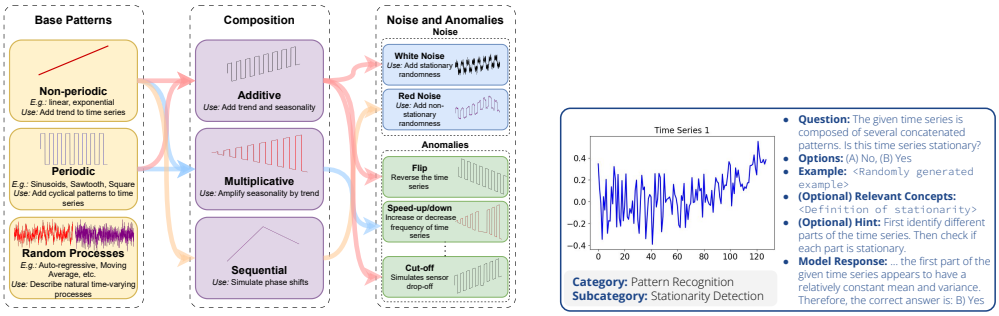


Figure 1: (*Left*) **Time Series Curation Pipeline:** The composition model generates controlled synthetic time series step-by-step. The pipeline enables diversity by combining different components to create numerous synthetic time series with varying properties. (*Right*) Each template evaluates a specific category, and includes a question, list of options, example question and answer pair for in-context learning, and optionally a hint and descriptions of complicated technical terms. Here, `GPT-4o` showcases its ability to transfer visual understanding and time series concepts into effective reasoning.

questions into simpler steps and textual descriptions of complicated technical terms. By incorporating these relevant concepts, we can isolate an LLM's ability to understand time series concepts (e.g., whether the mean and variance remain constant) from its understanding of complex technical jargon (e.g., stationarity). Each option (e.g. `"(A) Yes"`) is linked to a synthetic time series generator (Fig. 1)(*Left*) that produces a random time series as if the current option were true (e.g., a random stationary time series). This allows us to generate random but accurate time series at scale.

Generating Questions. We generate different questions from the same template by systematically varying the correct option and producing synthetic time series conditioned on the template and the correct option pair. Our simple and scalable approach, illustrated in Fig. 1(*Left*), involves sampling a small number of base patterns from a predefined pool and combining them using a composition function. Base patterns can be periodic (e.g., sine function), non-periodic (e.g., linear increasing function), or random time-varying processes (e.g., AR process). Depending on the template's nature, the final step adds additive noise or anomalies using the anomaly injection process described in (11).

Improving Questions Iteratively. We use Item Response Theory (IRT) (25) to achieve finer grained control over the quality of randomly generated questions included in the `TimeSeriesExam`. IRT is a statistical framework that models the relationship between an individual's (or LLM's) latent trait (e.g., knowledge, ability) and their responses to a set of items (e.g., questions on a test). It is a valuable tool in exam development as it helps to identify weak exam items, ensures consistent scoring across different versions of the exam, and also allows tailoring the testing experience to the LLM's abilities.

Our primary objective is to design a `TimeSeriesExam` where each question can maximally distinguish the abilities of candidate LLMs. We use the two-parameter logistic (2PL) model for this. Formally, for LLM j with ability θ_j, and question i with difficulty b_i, discrimination ability a_i, the 2PL model defines the probability of a correct response as: $\mathbb{P}(r_{ij} = 1|a_i, b_i, \theta_j) = 1/(1 + e^{-a_i(\theta_j - b_i)})$

Each `TimeSeriesExam` typically undergoes 1–3 rounds of iterative refinement. In each round, all candidate models take the exam. Based on their responses, we fit the parameters of Equation 3.1 using maximum likelihood estimation (MLE). Then, we drop $X\%$ of samples with the lowest sum of difficulty and discrimination ability. Finally, we randomly re-generate questions from the dropped templates. This iterative process is detailed in Algorithm 1 and the hyper-parameters of the fitting process are provided in App. B.2. We demonstrate the increase in the differentiability parameter across the iteration rounds and provide an analysis of the trajectory in App. B.3

While detailed empirical results are presented in Section 4, the PoC establishes two takeaways: (i) template-based generation yields diverse, controllable items across core reasoning categories, and

(ii) modern VLMs still struggle on higher-order time series reasoning, which motivates a scalable pipeline for real datasets with minimal expert time.

3.2 TIMESERIESEXAMAGENT: A SCALABLE DOMAIN-AGNOSTIC BENCHMARK CREATION TOOL

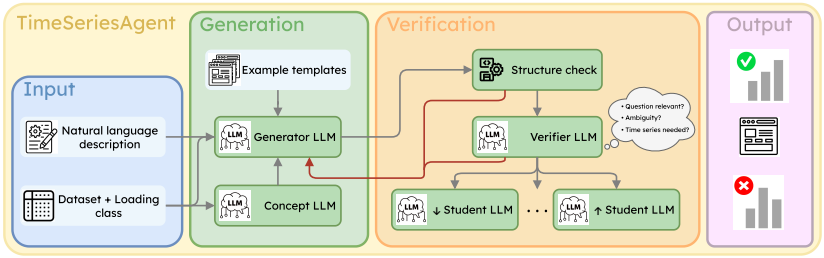


Figure 2: `TimeSeriesExamAgent` architecture. The user provides exam-making instructions and a custom dataset with minimal loading code. Agent outputs question templates – Python functions generated by a generator LLM and filtered through three progressive stages of verification (syntax and output format check, validation by LLM judge, capability-aligned filtering). Arrows denote data flow, red ones show direction for rejected templates.

Building on the proof-of-concept in the previous subsection that surfaced persistent gaps in LLMs' time series reasoning, we now focus on the practical need to evaluate models on domain-specific datasets at scale and with minimal expert effort. Domain experts are often interested in assessing LLMs on specialized reasoning capabilities rather than on broad, preexisting benchmarks (e.g., evaluating anomaly detection in ECG data versus generic healthcare reasoning). To this end, they typically possess domain-specific datasets and wish to construct benchmarks that reflect the reasoning challenges within these datasets. However, building such benchmarks manually is labor-intensive, as we have demonstrated with `TimeSeriesExam`. To address this challenge, we propose `TimeSeriesExamAgent`, a multi-agent framework that combines planning, generation, and verification to enable automatic benchmark construction on real datasets while minimizing expert time.

Setup An overview of the agentic framework is shown in Fig. 2. The Generation Agent takes as input a description of the natural language task T and a data set D. The description T may include user guidelines for generation, contextual information about the dataset, or other relevant instructions. For convenience, we denote each sample in D as (x_i, z_i), where $x_i \in \mathbb{R}^{n \times d}$ is a time series with n observations and d variables, and z_i is an auxiliary array containing metadata or labels related to the series. The user provides a dataset class D that supports basic operations such as querying the i-th sample.

Generation Motivated by `TimeSeriesExam`, we generate question templates instead of samples directly, as shown in Fig. 3. Templates offer two advantages: they are scalable, and their abstraction adds an extra layer of robustness. By relying on structured, rule-based generation rather than manual inputs, they reduce the chance of human errors or inconsistencies. Our generator LLM produces a predefined number of templates, each implemented as a Python function. A template contains a formatted string for the question and options, together with parameters that control how many questions to generate. For each question, the template samples a pair (x_i, z_i) from the dataset D and applies a rule-based calculation to determine the correct answer from the time series. For example, in a trend-detection template, the function computes the linear trend coefficient of x_i and selects "Yes, there is a linear trend" if the coefficient exceeds a specified threshold. In addition to such signal-derived logic, templates can also utilize the auxiliary property z_i, effectively transforming classification problems into question–answer form. For instance, if an ECG series in the dataset is labeled as exhibiting atrial fibrillation, the template can present this label as one of the multiple-choice options. Each generated sample consists of the question, its options, the correct answer, and one or more associated time series represented as numerical values. We provide a breakdown of the Generation Agent and its prompt in App. C. An example template is also provided.

Verification We observe that LLM-based generation frequently produces errors or irrelevant outputs, motivating the need for a structured verification process. We propose a multistage verification process to check the accuracy and relevance of each template. If a template fails at any stage, it is returned to the generation agent with feedback. The generation is iterative with a maximum of three attempts, after which the ongoing template is discarded to avoid excessive context length and cost from repeated failures.

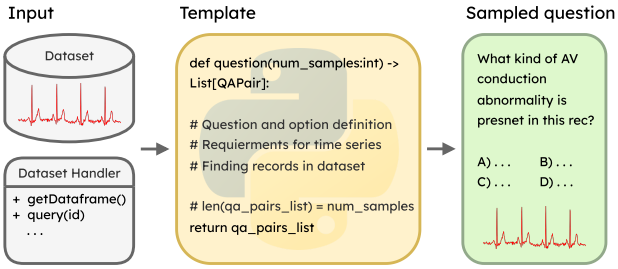


Figure 3: Question generation process: With information about dataset, `TimeSeriesExamAgent` generates question template in a form of Python functions. The created function can be called to get arbitrary number of question samples.

Structure verification We check whether the generated template can be executed successfully. We execute the generated template $k = 3$ times; if there are any failures, the error message is returned as feedback.

Content verification Certain aspects of quality control are particularly well-suited for using LLM-as-a-judge evaluation. For example, verifying that a question is grammatically correct, free of ambiguity or bias, and genuinely answerable from the accompanying time series can be effectively handled by an LLM. To this end, we use an LLM verifier to assess the validity of each template. We use a binary scheme: a template must pass all categories to be accepted. Any failure triggers rejection and regeneration, ensuring robustness. Details are in App. D.

Capability-Aligned Filtering Inspired by `TimeSeriesExam`, which leverages IRT to enhance differentiability across questions, we adoptted a similar but localized framework that operates at the level of each template. To detect templates that generate overly simple or irrelevant exams, we evaluate them using a set of test-taking LLMs with varying capabilities. This approach is supported bmy educational theory, particularly the expertise reversal effect (18). A template is discarded if weaker LLMs achieve higher average accuracy than stronger models, as this typically indicates that the template is flawed or noisy rather than genuinely discriminative. Templates are retained if performance scales with model capability, or if all models perform poorly, since such questions may still capture genuine difficulty. We provide hyperparameters in App. F and other design specifics in App. E

4 Experimental Setup, Results and Analysis

4.1 TimeSeriesExam Evaluation

Model	Similarity	Pattern	Causality	Noise	Anomaly	Overall
`Gemma-3-27B-IT`(36)	0.62	0.59	0.51	0.71	0.51	0.59
`GPT-4o`(15)	**0.78**	**0.78**	**0.61**	**0.77**	0.54	**0.73**
`Qwen2.5-VL-72B`(2)	0.60	0.45	0.49	0.40	0.22	0.44
`Gemini-2.5-Pro`(9)	**0.78**	0.76	0.57	0.65	**0.59**	0.71

Table 3: Accuracy of model on each category of `TimeSeriesExam`.

Table 3 reports the accuracy on `TimeSeriesExam` in five categories of reasoning. We evaluated several state-of-the-art vision language models (VLMs) following the protocol described in App. H. Overall, GPT-4o achieves the strongest performance, closely followed by Gemini-2.5-Pro, while Qwen2.5-VL lags significantly behind.

Across categories, models perform best on relatively shallow tasks such as similarity and pattern recognition, where surface-level cues often suffice. Performance drops sharply for more challenging

categories. In particular, anomaly detection proves to be the most challenging, reflecting the need to integrate subtle statistical deviations with contextual reasoning. These results highlight that, while modern VLMs capture basic time series patterns, they fall short on higher-order reasoning tasks. We provide a case study in App. B.5

4.2 TIMESERIESEXAMAGENT GENERATED SAMPLE EVALUATION

First, we generate one exam for each of the five real world datasets: PTB-XL (38), MIT-BIH (29), MIMIC-IV Waveform (28), yahoo finance stock dataset (35), and WeatherBench 2 (34). In total, we have 209 samples for YFinance, 197 samples for MIT-BIH, 151 samples for PTB-XL, 205 samples for MIMIC-IV Waveform, and 95 samples for WeatherBench 2. We sample 4 or 5 instances per template. Thus, the difference in the number of generated samples is a result of the template filtering mechanism above.

Model	Dataset					Average
	MIT-BIH	PTB-XL	MIMIC-IV W	YFinance	WeatherBench2	
gpt-4o (15)	0.416	0.424	0.385	**0.586**	**0.389**	0.440
o3-mini (33)	0.442	0.477	0.356	0.555	0.379	0.442
Qwen2.5-VL-Instruct (2)	0.411	0.490	**0.439**	0.572	0.368	**0.456**
Gemma-3-27b-it (36)	**0.497**	**0.517**	0.370	0.534	0.232	0.430

Table 4: Comparative performance of four vision–language models across medical (MIT-BIH, PTB-XL, MIMIC-IV Waveform (MIMIC-IV W)), financial (YFinance), and meteorological (WeatherBench 2) time series datasets. The results highlight dataset-specific strengths; nonetheless, all models achieve less than 55 mean accuracy, underscoring the difficulty of time series reasoning for current VLMs. The evaluation protocol is provided in App. H

We select candidate models to cover a diverse range of performance levels, as indicated by the OpenVLM Leaderboard (10). In Table 4, we find that while general-purpose multimodal models such as gpt-4o perform well on finance-related questions, their performance is weaker on healthcare benchmarks. This contrast could suggest that the general reasoning ability does not always transfer across domains, particularly when tasks require domain-specific expertise or fine-grained interpretation of physiological signals.

4.3 TIMESERIESEXAMAGENT GENERATES QUESTIONS WITH DIVERSITY COMPARABLE TO HUMAN-CURATED BENCHMARKS

We evaluate the diversity of questions generated by our framework against ECG-QA (31), a template-based benchmark built on PTB-XL. Our aim is to show that TimeSeriesExamAgent achieves comparable variety without manual template design. For each benchmark, we randomly sampled 50 questions and computed pairwise embedding distances. Embeddings were extracted using Qwen/Qwen3-Embedding-8B[1], the top open-source model on the Hugging Face MTEB leaderboard[2].

Benchmark Dataset	Mean ± Std	
	Embedding	Normalized Levenshtein
ECG-QA	0.207 ± 0.079	0.519 ± 0.157
TimeSeriesExamAgent (ours)	**0.301 ± 0.070**	**0.542 ± 0.039**

Table 5: Diversity of questions measured by embedding and normalized Levenshtein distances. Higher values indicate greater variability in phrasing.

As shown in Table 5, our framework achieves a level of diversity that is broadly comparable to human-curated benchmarks. This suggests that it can capture a range of question formulations

[1] https://huggingface.co/Qwen/Qwen3-Embedding-8B

[2] https://huggingface.co/spaces/mteb/leaderboard

without relying on handcrafted templates, which may help its scalability to other domains. For completeness, we include a visualization in App. G.2 to further illustrate this observation.

Table 6: **Combined Quality Evaluation**. Scores (1–10) averaged across four criteria. Rows are grouped by their original source tables.

Dataset	Mean Result			
	Specificity	Unambiguity	Domain Relevance	Answerability
Finance Domain				
FinMME	8.10	**7.59**	6.62	6.95
MTBench	6.88	6.11	8.35	7.29
TimeSeriesExamAgent (ours)	**8.29**	7.24	**8.89**	**8.57**
Medicine Domain				
ECG-QA	5.60	5.77	8.17	8.47
TimeSeriesExamAgent (ours)	**8.43**	**8.40**	**9.00**	**9.10**

We employed an LLM-as-a-jury approach using G-Eval, where a panel of models (Gemini-1.5-Pro, GPT-3.5-Turbo, and Qwen2.5-VL-72B-Instruct) evaluated the quality of each question. To ensure cost efficiency, we selected relatively weaker models, as prior work shows this setup can maintain evaluation quality while mitigating intra-model bias (37). Each model independently assigned a score from 1 to 10 based on four criteria. The aggregated results, reported in Table 6, show that TimeSeriesExamAgent outperforms ECG-QA across all dimensions, particularly in specificity and answerability. This indicates that our framework generates precise, well-grounded, and domain-appropriate questions.

4.4 LLMs TRAINED ON OUR GENERATED SAMPLES EXHIBIT TRANSFERABLE REASONING SKILLS ON ESTABLISHED DATASETS

Another way to assess the value of TimeSeriesExamAgent is to test whether its generated data supports transfer learning. We finetuned VLM Qwen2.5-VL-3B-Instruct. We first generated 2000 training samples using TimeSeriesExam based on the PTB-XL dataset, while testing was conducted on 12000 randomly selected samples from the ECG-QA (32) test split using MIMIC-IV data. Training parameters are provided in App. I.

Method	Accuracy	
	General	Parsable
Random answering	34.9%	34.9%
Base	21.8%	34.6%
Fine-tuned	**47.0%**	**49.7%**

Table 7: Results of VLM fine-tuning on the exams generated by TimeSeriesExamAgent. *General*: all incorrectly formatted responses are treated as wrong answers. *Parsable*: only correctly formatted responses are evaluated.

Table 7 shows clear gains under the strict accuracy metric: the unfine-tuned Base model achieves 21.8%, while fine-tuning on TimeSeriesExamAgent–generated exams lifts accuracy to 47.0%, which gives 216% relative improvement. The fine-tuned model also surpasses the Random baseline (34.9%), indicating that agent-generated questions provide genuinely useful supervision rather than superficial patterning. Overall, these results suggest that synthetic, agent-curated exams can improve decision quality.

5 DISCUSSION, OPEN QUESTIONS AND OPPORTUNITIES

Reliance on Expert-Generated Prompts A key limitation of TimeSeriesExamAgentis that the quality of the generated exams ultimately depends on the user instruction and coverage of the underlying time series dataset. For example, if important clinical instructions for the healthcare data set are absent, the resulting questions may not adequately capture the reasoning challenges faced in practice. Therefore, domain specialists are still required to provide carefully designed prompts that specify the scope and relevance of the questions. In offline sessions with cardiologists, we observed that when clinicians contributed targeted feedback during the prompt design stage, the resulting exams were consistently judged as more clinically valid and useful J. This highlights the importance

of structured collaboration between automated systems and human experts, especially in high-stakes domains such as healthcare. We aim to provide a quantitative analysis in the rebuttal phase.

Demand for human-in-the-loop evaluation. Building on the previous observation, we integrated optional human-in-the-loop modules into `TimeSeriesExamAgent` to facilitate a more practical deployment. These modules allow domain experts to refine templates, validate generated questions, and iteratively improve exam quality. Although we received encouraging anecdotal feedback from clinicians and practitioners who interacted with the system, the influence of such human feedback pipelines could not be systematically tested within the scope of this study. A formal evaluation of how human involvement impacts benchmark validity and downstream model assessment remains an important direction for future work and the community.

Limitation for using LLM-as-a-judge for quality control. To reduce reliance on manual review, we employed LLM-as-a-judge strategies to evaluate the quality of generated exam items. Specifically, we used verifier models to assess whether questions were grammatically correct, free from ambiguity, and answerable given the associated time series. In addition, we applied capability-aligned filtering, discarding templates that failed to discriminate between stronger and weaker LLMs. While this approach significantly improved scalability and reduced manual overhead, it also raises open questions about the reliability and biases of LLM-based evaluators themselves. Developing more rigorous evaluation protocols—potentially combining automated scoring with selective human review—remains an important challenge, as manual evaluation of this framework often involves human expert.

Limited Evaluation Mode. In the current framework, questions are evaluated by providing the time series as image input. This design introduces several challenges. First, as the number of channels in a time series increases, the resolution and readability of the image naturally deteriorate, making it harder for models to extract relevant information. Second, certain question types—such as those requiring explicit computation of a metric before making a decision—are particularly difficult to answer from images alone. By contrast, other questions may be more effectively represented in textual form rather than as visual plots. These challenges highlight the need for an intelligent decision-making tools, such as an agentic framework, to dynamically choose the most suitable representation. There is growing interest in agentic frameworks for time series analysis tasks (6; 47). Our benchmark provides a natural testbed for such systems, since many of the generated questions require multi-step reasoning, or direct computation over numeric data. Enabling agentic AI systems to autonomously write and execute code in order to answer our benchmark questions would provide valuable insights into their reasoning fidelity and robustness. Beyond evaluating LLMs passively, this opens the door to studying interactive and tool-augmented agents, bridging the gap between static reasoning benchmarks and real life decision making scenarios.

Natural extension beyond time series. Although we focus on time series data, the underlying framework is not inherently restricted to this modality. Our design only assumes access to structured data and domain-specific prompts, making it extensible to other settings such as images, tables, or even multimodal combinations of signals and text. We chose time series as a starting point because it is a highly structured domain with well-established industrial applications.

6 Conclusion

This work first examined whether LLMs can reason about fundamental time-series concepts. To address these questions, we introduced `TimeSeriesExam`, a controlled benchmark for probing conceptual understanding, and `TimeSeriesExamAgent`, a scalable framework that enables practitioners to generate customized benchmarks from their own data. Our experiments show that while LLMs capture some surface-level patterns, they continue to struggle with more complex reasoning such as anomaly detection. At the same time, benchmarks generated by `TimeSeriesExamAgent` match or exceed the diversity and quality of human-curated datasets, and can even provide useful finetuning signals for downstream tasks. These results suggest that automated, agentic benchmark construction can help make evaluation more adaptive and domain-relevant.

Reproducibility Statement

We release all evaluated datasets at `https://anonymous.4open.science/r/TimeSeriesExamAgentSubmission-C9C5`. The evaluation protocol is described in App. H, and the appendix further provides implementation details of the pipeline, including prompts and configuration settings.

References

[1] Abdul Fatir Ansari, Lorenzo Stella, Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Sundar Rangapuram, Sebastian Pineda Arango, Shubham Kapoor, et al. Chronos: Learning the language of time series. *arXiv preprint arXiv:2403.07815*, 2024.

[2] Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-vl technical report. *arXiv preprint arXiv:2502.13923*, 2025.

[3] Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The long-document transformer, 2020.

[4] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[5] Natasha Butt, Varun Chandrasekaran, Neel Joshi, Besmira Nushi, and Vidhisha Balachandran. Benchagents: Automated benchmark creation with agent interaction. *arXiv preprint arXiv:2410.22584*, 2024.

[6] Yifu Cai, Xinyu Li, Mononito Goswami, Michał Wiliński, Gus Welter, and Artur Dubrawski. Timeseriesgym: A scalable benchmark for (time series) machine learning engineering agents. *arXiv preprint arXiv:2505.13291*, 2025.

[7] Defu Cao, Furong Jia, Sercan O Arik, Tomas Pfister, Yixiang Zheng, Wen Ye, and Yan Liu. Tempo: Prompt-based generative pre-trained transformer for time series forecasting. *arXiv preprint arXiv:2310.04948*, 2023.

[8] Jialin Chen, Aosong Feng, Ziyu Zhao, Juan Garza, Gaukhar Nurbek, Cheng Qin, Ali Maatouk, Leandros Tassiulas, Yifeng Gao, and Rex Ying. Mtbench: A multimodal time series benchmark for temporal reasoning and question answering. *arXiv preprint arXiv:2503.16858*, 2025.

[9] Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.

[10] Haodong Duan, Junming Yang, Yuxuan Qiao, Xinyu Fang, Lin Chen, Yuan Liu, Xiaoyi Dong, Yuhang Zang, Pan Zhang, Jiaqi Wang, et al. Vlmevalkit: An open-source toolkit for evaluating large multi-modality models. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pages 11198–11201, 2024.

[11] Mononito Goswami, Cristian Challu, Laurent Callot, Lenon Minorics, and Andrey Kan. Unsupervised model selection for time-series anomaly detection. *arXiv preprint arXiv:2210.01078*, 2022.

[12] Mononito Goswami, Konrad Szafer, Arjun Choudhry, Yifu Cai, Shuo Li, and Artur Dubrawski. Moment: A family of open time-series foundation models. *arXiv preprint arXiv:2402.03885*, 2024.

[13] Clive WJ Granger. Investigating causal relations by econometric models and cross-spectral methods. *Econometrica: journal of the Econometric Society*, pages 424–438, 1969.

[14] Gauthier Guinet, Behrooz Omidvar-Tehrani, Anoop Deoras, and Laurent Callot. Automated evaluation of retrieval-augmented language models with task-specific exam generation. *arXiv preprint arXiv:2405.13622*, 2024.

[15] Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

[16] Aya Abdelsalam Ismail, Mohamed Gunady, Hector Corrada Bravo, and Soheil Feizi. Benchmarking deep learning interpretability in time series predictions. *Advances in neural information processing systems*, 33:6441–6452, 2020.

[17] Ming Jin, Shiyu Wang, Lintao Ma, Zhixuan Chu, James Y Zhang, Xiaoming Shi, Pin-Yu Chen, Yuxuan Liang, Yuan-Fang Li, Shirui Pan, et al. Time-llm: Time series forecasting by reprogramming large language models. *arXiv preprint arXiv:2310.01728*, 2023.

[18] Slava Kalyuga. Expertise reversal effect and its implications for learner-tailored instruction. *Educational psychology review*, 19(4):509–539, 2007.

[19] 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. *arXiv preprint arXiv:2503.01875*, 2025.

[20] John Patrick Lalor and Pedro Rodriguez. py-irt: A scalable item response theory library for python. *INFORMS Journal on Computing*, 35(1):5–13, 2023.

[21] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474, 2020.

[22] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks. *arXiv preprint arXiv:2005.11401*, 2020.

[23] Haoxin Liu, Shangqing Xu, Zhiyuan Zhao, Lingkai Kong, Harshavardhan Prabhakar Kamarthi, Aditya Sasanur, Megha Sharma, Jiaming Cui, Qingsong Wen, Chao Zhang, et al. Time-mmd: Multi-domain multimodal dataset for time series analysis. *Advances in Neural Information Processing Systems*, 37:77888–77933, 2024.

[24] Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. G-eval: Nlg evaluation using gpt-4 with better human alignment. *arXiv preprint arXiv:2303.16634*, 2023.

[25] Frederic M Lord and Melvin R Novick. *Statistical theories of mental test scores*. IAP, 2008.

[26] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[27] Mike A Merrill, Mingtian Tan, Vinayak Gupta, Tom Hartvigsen, and Tim Althoff. Language models still struggle to zero-shot reason about time series. *arXiv preprint arXiv:2404.11757*, 2024.

[28] Benjamin Moody, Shaoxiong Hao, Benjamin Gow, Tom Pollard, Weixuan Zong, and Roger Mark. Mimic-iv waveform database. *PhysioNet*, 2022.

[29] George B Moody and Roger G Mark. The impact of the mit-bih arrhythmia database. *IEEE engineering in medicine and biology magazine*, 20(3):45–50, 2001.

[30] Jesse Mu, Xiang Lisa Li, and Noah Goodman. Learning to compress prompts with gist tokens, 2024.

[31] Jungwoo Oh, Gyubok Lee, Seongsu Bae, Joon-myoung Kwon, and Edward Choi. Ecg-qa: A comprehensive question answering dataset combined with electrocardiogram. *Advances in Neural Information Processing Systems*, 36:66277–66288, 2023.

[32] Jungwoo Oh, Gyubok Lee, Seongsu Bae, Joon-myoung Kwon, and Edward Choi. Ecg-qa: A comprehensive question answering dataset combined with electrocardiogram. *Advances in Neural Information Processing Systems*, 36, 2024.

[33] OpenAI. Openai o3-mini system card. `https://openai.com/index/o3-mini-system-card/`, January 2025. Accessed: 2025-08-22.

[34] Stephan Rasp, Stephan Hoyer, Alexander Merose, Ian Langmore, Peter Battaglia, Tyler Russel, Alvaro Sanchez-Gonzalez, Vivian Yang, Rob Carver, Shreya Agrawal, Matthew Chantry, Zied Ben Bouallegue, Peter Dueben, Carla Bromberg, Jared Sisk, Luke Barrington, Aaron Bell, and Fei Sha. Weatherbench 2: A benchmark for the next generation of data-driven global weather models, 2023.

[35] Ranan Roussi. yfinance: Yahoo! finance market data downloader. `https://github.com/ranaroussi/yfinance`, 2017. Accessed: 2025-08-22.

[36] Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, et al. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*, 2025.

[37] Pat Verga, Sebastian Hofstatter, Sophia Althammer, Yixuan Su, Aleksandra Piktus, Arkady Arkhangorodsky, Minjie Xu, Naomi White, and Patrick Lewis. Replacing judges with juries: Evaluating llm generations with a panel of diverse models. *arXiv preprint arXiv:2404.18796*, 2024.

[38] Patrick Wagner, Nils Strodthoff, Ralf-Dieter Bousseljot, Dieter Kreiseler, Fatima I Lunze, Wojciech Samek, and Tobias Schaeffter. Ptb-xl, a large publicly available electrocardiography dataset. *Scientific data*, 7(1):1–15, 2020.

[39] Qingyue Wang, Yanhe Fu, Yanan Cao, Shuai Wang, Zhiliang Tian, and Liang Ding. Recursively summarizing enables long-term dialogue memory in large language models, 2025.

[40] Wanying Wang, Zeyu Ma, Pengfei Liu, and Mingang Chen. Testagent: A framework for domain-adaptive evaluation of llms via dynamic benchmark construction and exploratory interaction. *arXiv preprint arXiv:2410.11507*, 2024.

[41] Xingyao Wang, Boxuan Li, Yufan Song, Frank F Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, et al. Openhands: An open platform for ai software developers as generalist agents. *arXiv preprint arXiv:2407.16741*, 2024.

[42] Xu Wang, Jiaju Kang, Puyu Han, Yubao Zhao, Qian Liu, Liwenfei He, Lingqiong Zhang, Lingyun Dai, Yongcheng Wang, and Jie Tao. Ecg-expert-qa: A benchmark for evaluating medical large language models in heart disease diagnosis. *arXiv preprint arXiv:2502.17475*, 2025.

[43] Yilin Wang, Peixuan Lei, Jie Song, Yuzhe Hao, Tao Chen, Yuxuan Zhang, Lei Jia, Yuanxiang Li, and Zhongyu Wei. Itformer: Bridging time series and natural language for multi-modal qa with large-scale multitask dataset. *arXiv preprint arXiv:2506.20093*, 2025.

[44] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

[45] Andrew Robert Williams, Arjun Ashok, Étienne Marcotte, Valentina Zantedeschi, Jithendaraa Subramanian, Roland Riachi, James Requeima, Alexandre Lacoste, Irina Rish, Nicolas Chapados, et al. Context is key: A benchmark for forecasting with essential textual information. *arXiv preprint arXiv:2410.18959*, 2024.

[46] Gerald Woo, Chenghao Liu, Doyen Sahoo, Akshat Kumar, and Steven Hoi. Cost: Contrastive learning of disentangled seasonal-trend representations for time series forecasting. *arXiv preprint arXiv:2202.01575*, 2022.

[47] Wen Ye, Wei Yang, Defu Cao, Yizhou Zhang, Lumingyuan Tang, Jie Cai, and Yan Liu. Domain-oriented time series inference agents for reasoning and automated analysis. *arXiv preprint arXiv:2410.04047*, 2024.

[48] Tian Zhou, Peisong Niu, Liang Sun, Rong Jin, et al. One fits all: Power general time series analysis by pretrained lm. *Advances in neural information processing systems*, 36:43322–43355, 2023.

[49] Nina Żukowska, Mononito Goswami, Michał Wiliński, Willa Potosnak, and Artur Dubrawski. Towards long-context time series foundation models. *arXiv preprint arXiv:2409.13530*, 2024.

A TIMESERIESEXAM DATASET DETAILS

Meta-information Type	Anomaly Detection	Similarity Analysis	Noise Understanding	Pattern Recognition	Causality Analysis
# Questions	129	113	87	371	63
%age questions with 2 time series	31.01	100	14.94	3.77	100

Table 8: TimeSeriesExam meta-information breakdown for each category. Each question is associated with a time series of length 128 time steps, and an example time series of length 64 time steps.

B TIMESERIESEXAM ALGORITHMS AND PARAMETERS

B.1 ITERATIVE REFINEMENT ALGORITHM

Algorithm 1 Iterative Dataset Refinement with IRT and Resampling

Require: num_iterations = 3, drop_percentage = 0.2, initial dataset D_0

$D \leftarrow D_0$
for iteration $= 1$ to num_iterations **do**
  Evaluate each candidate i on D, and obtain the response set $R = \{r_{ij} \mid r_{ij} = 1$ if candidate i correctly answers question $j\}$
  Fit the IRT model to obtain the discrimination parameters $\mathbf{A} = \{a_j \mid j \in \text{Questions}\}$ and difficiulty parameter $\mathbf{B} = \{b_j \mid j \in \text{Questions}\}$
  Normalize set $\mathbf{A}$ and $\mathbf{B}$ between 0 and 1, and calculate score $\mathbf{S} = \{b_j + a_j \mid j \in \text{Questions}\}$
  Find $\mathbf{S}'$ which is the score for samples that are answered correctly by the best model in the round
  Find the index set $I = \{j \mid a_j < \text{Quantile}(\mathbf{S}', \text{drop\_percentage})\}$, where a_j is less than the drop_percentage quantile of $\mathbf{A}$
  for each $j \in I$ **do**
    Resample a new question q' from the same category as question j
    Set $D[j] \leftarrow q'$
  end for
end for
return D

B.2 IRT MODEL PARAMETERS

The IRT models are fitted using library *py-irt* (20). The parameters are epochs=2000, lr=0.1, lrdecay=0.9999, dropout=0.5, hidden=100

B.3 AVERAGE SAMPLE DISCRIMINATION PARAMETER OVER ROUNDS

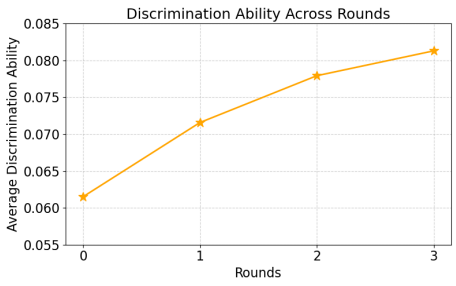


Figure 4: The sample average discrimination parameter across rounds shows an upward trend, indicating an improved ability of the questions to differentiate candidates with varying levels of ability.

B.4 DROPPED DATASET DISTRIBUTION PER ROUND

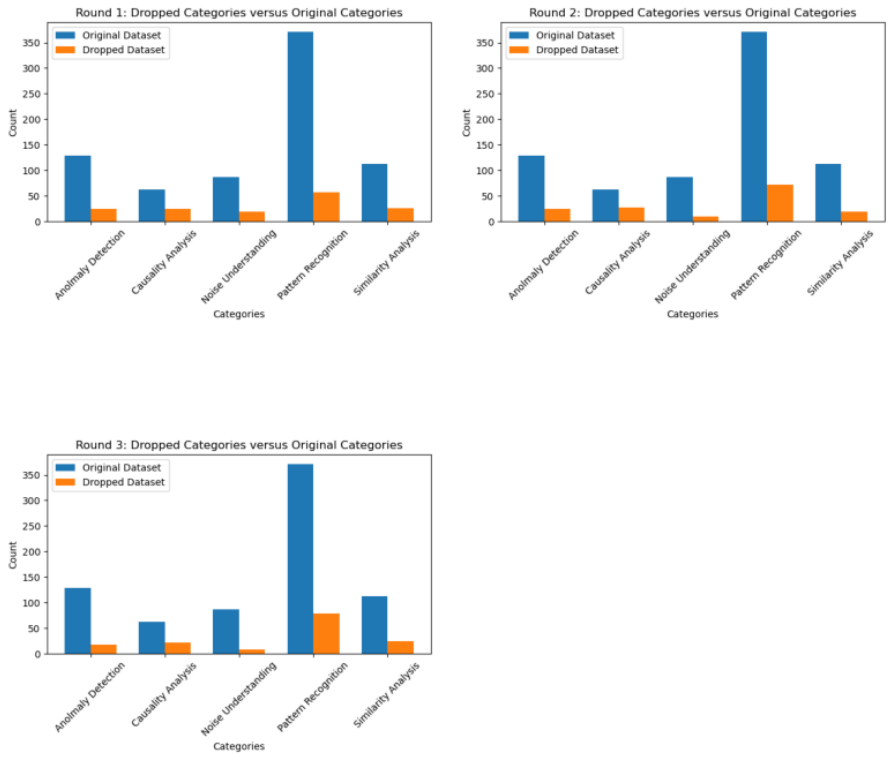
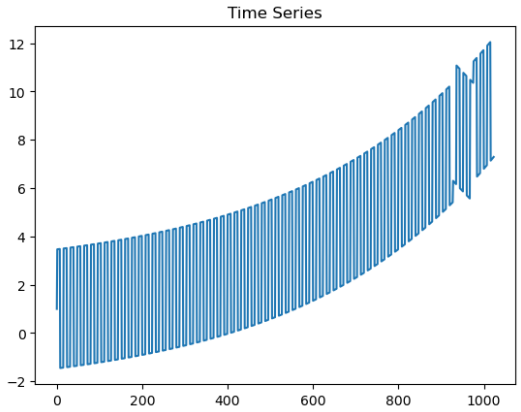


Figure 5: Dropped Dataset Distribution per round. Dropped category distribution per round generally mirrors the overall category distribution.

We can observe in Fig. 5 that the proportion of dropped questions for each category is approximately uniform. The difference in number of questions in each category is a result of different template curated.

B.5 Case Study 1: Anolmay pattern disrupts visual reasoning



GPT-4o

Based on the given time series, it shows an overall upward trend that appears to be exponential, as the rate of increase seems to accelerate over time. Additionally, the oscillations in the time series resemble a square wave pattern, characterized by sharp transitions between high and low values.
Answer: B) Exponential trend and square wave

Gemma-3-27B-IT

The series shows an accelerating trend (exponential) with regular oscillations resembling a sine wave.
Answer: A) Exponential trend and sine wave

Qwen2.5-VL-72B

The overall upward pattern is consistent with a linear trend. The cyclic component matches a sine wave.
Answer: A) Linear trend and sine wave

Gemini-2.5-Pro

The trend curves upward, suggesting exponential. Oscillations are smooth and sinusoidal.
Answer: A) Linear trend and sine wave

This example shows common failure modes in time-series reasoning. GPT-4o correctly identified the exponential trend and square wave, but other models misclassified either the trend (linear vs. exponential) or the cyclic component (sine vs. square). Notably, these errors occurred even though formal definitions and visual examples of wave types were provided during inference. Models often defaulted to smoother structures like sine waves when uncertain, suggesting reliance on heuristics rather than careful inspection.

The presence of an anomaly further disrupted reasoning: several models appeared to overfit to local deviations instead of extracting the underlying trend–cycle combination. This highlights a broader limitation. Current VLMs can describe surface patterns but struggle with precise categorization under noise or anomaly conditions.

C TIMESERIESEXAMAGENT GENERATION AGENT WORKFLOW

We rely on two stages of generation for the templates: planning and generating, inspired by the chain-of-thought (CoT) prompting(44).

Generation planning To provide a relevant and diverse set of templates, we rely on a comprehensive list of domain-specific concepts. There are several ways our pipeline generates a list of concepts:

1. LLM generation: User guidelines and dataset descriptions are provided as input to an LLM, which proposes the concepts.
2. Web Search: We provide the option for generator LLM obtain concepts through web search.
3. Retrieval Augmented Generation: As an option, the user could also provide a relevant file from which the LLM reads and generates concepts(21).

Template generation As input to our generator, the following components are provided:

- User-provided guidelines: a document containing the user's goal or specific requirements,
- Dataset description: a list of columns and example values with ranges from the dataset, with a short usage example,
- List of concepts: generated in previous step. For each template, our pipeline will choose a concept at random to ensure diversity.
- Example templates[Optional]: user-provided few-shot examples presenting required structural elements (4).

C.1 GENERATION PROMPT

```
Here is the goal of the exam questions:
{user_info_text}

Here are sample concepts on which you can base your question
   generation:
{concept_conversation}

Use the concept numbered {concept_no} from the list to guide the
   design of your question template.

Here is the description of the dataset you will use to generate
   the question:
{dataset_describe}

In your template, use the provided `user_dataset` object. Use its
   `query(index)` method to load relevant time series data.

Do not select time series randomly. First, formulate the question,
    and then choose a time series that fits its logic and
   reasoning needs.

Generate one function-based question template now.
```

C.2 EXAMPLE OF QUESTION TEMPLATE

```
def question_6(num_samples, verbose=False):
    hyperparameters = {
        "min_trend_days": 20,
        "max_series_length": 3000,
        "trend_strength_threshold": 0.7,
```

```
    "momentum_window": 10,
}

question = "Analyzing the price movements of {ticker} over the
 given time period, does the price trend demonstrate strong
momentum and sustainability, or does it show signs of weakness
 and potential reversal?"

options = [
    "The trend shows strong momentum with consistent
directional movement and minimal pullbacks, suggesting the
trend is likely to continue.",
    "The trend shows signs of weakness with frequent reversals
 and inconsistent momentum, suggesting a potential trend
change.",
    "The trend shows mixed signals with alternating periods of
 strength and weakness, making direction unclear.",
    "The price movement shows no clear trend pattern,
indicating a ranging or sideways market."
]

def calculate_trend_strength(prices):
    if len(prices) < hyperparameters["min_trend_days"]:
        return None, None

    returns = np.diff(prices) / prices[:-1]

    # Calculate momentum consistency
    positive_days = np.sum(returns > 0)
    negative_days = np.sum(returns < 0)
    total_days = len(returns)

    directional_consistency = max(positive_days, negative_days
) / total_days

    # Calculate average magnitude of moves
    avg_abs_return = np.mean(np.abs(returns))

    # Calculate trend persistence (consecutive moves in same
direction)
    consecutive_moves = []
    current_streak = 1
    for i in range(1, len(returns)):
        if np.sign(returns[i]) == np.sign(returns[i-1]):
            current_streak += 1
        else:
            consecutive_moves.append(current_streak)
            current_streak = 1
    consecutive_moves.append(current_streak)

    avg_streak = np.mean(consecutive_moves)
    max_streak = max(consecutive_moves)

    # Determine overall trend direction
    overall_return = (prices[-1] - prices[0]) / prices[0]
    trend_direction = "up" if overall_return > 0 else "down"

    return {
        "directional_consistency": directional_consistency,
```

```
            "avg_abs_return": avg_abs_return,
            "avg_streak": avg_streak,
            "max_streak": max_streak,
            "overall_return": abs(overall_return),
            "trend_direction": trend_direction
        }, returns

    qa_pairs = []
    df = user_dataset.get_dataframe()

    attempted_tickers = set()

    while len(qa_pairs) < num_samples:
        if verbose:
            print(f"[Question 6] Generating question {len(qa_pairs
)} / {num_samples}")

        # Select a ticker that hasn't been attempted
        available_tickers = [i for i in df.index if i not in
attempted_tickers]
        if not available_tickers:
            break

        ticker_id = random.choice(available_tickers)
        attempted_tickers.add(ticker_id)

        ticker = df.loc[ticker_id, 'ticker']
        prices = user_dataset.query(ticker_id)

        if len(prices) < hyperparameters["min_trend_days"]:
            continue

        # Limit series length
        if len(prices) > hyperparameters["max_series_length"]:
            start_idx = random.randint(0, len(prices) -
hyperparameters["max_series_length"])
            prices = prices[start_idx:start_idx + hyperparameters
["max_series_length"]]

        # Select a subset for analysis (to make question more
focused)
        analysis_length = min(len(prices), random.randint(50, 200)
)
        start_idx = random.randint(0, len(prices) -
analysis_length)
        analysis_prices = prices[start_idx:start_idx +
analysis_length]

        trend_metrics, returns = calculate_trend_strength(
analysis_prices)
        if trend_metrics is None:
            continue

        # Determine answer based on trend strength metrics
        strength_score = (
            trend_metrics["directional_consistency"] * 0.4 +
            min(trend_metrics["avg_streak"] / 5, 1.0) * 0.3 +
            min(trend_metrics["overall_return"] * 10, 1.0) * 0.3
        )
```

```
        if strength_score >= hyperparameters["
    trend_strength_threshold"] and trend_metrics["max_streak"] >=
    5:
            answer = options[0]
        elif strength_score < 0.4 or trend_metrics["
    directional_consistency"] < 0.6:
            answer = options[1]
        elif 0.4 <= strength_score < hyperparameters["
    trend_strength_threshold"]:
            answer = options[2]
        else:
            answer = options[3]

        question_text = question.format(ticker=ticker)

        qa_pairs.append({
            "question": question_text,
            "options": options,
            "answer": answer,
            "ticker": ticker,
            "ts": analysis_prices,
            "relevant_concepts": ["Volume-Price Trend Correlation
    ", "Trend Strength Analysis", "Price Momentum"],
            "domain": "finance",
            "detractor_types": ["Incorrect trend interpretation",
    "Misunderstanding momentum signals"],
            "question_type": "multiple_choice",
            "format_hint": "Please answer the question and provide
     the correct option letter, e.g., [A], [B], [C], [D], and
    option content at the end of your answer. All information need
     to answer the question is given. If you are unsure, please
    provide your best guess.",
        })

    return qa_pairs
```

C.3 Example of Natural Language Description

```
I want to create time series exam testing model understanding of
   finance time series data.

To load the data, use the provided ```user_dataset``` object.

Given time series come from Yahoo Finance, include closing price
   of a stock. Interval between samples is 1 day.
Make sure that the length of time series (total number of samples
   of one or two time series) does not excide 3000.

Please make sure that exams cannot be answer without timeseries!
```

C.4 Examples of Generated Questions

ECG Question Example

Q: Analyze the P-wave morphology and amplitude characteristics in this recording. What atrial abnormality is present?

A. RAO/RAE: Right atrial overload/enlargement with prominent P-waves
B. LAO/LAE: Left atrial overload/enlargement with bifid P-waves
C. Normal P-wave morphology with no atrial abnormalities
D. Absent P-waves indicating atrial fibrillation

answer: LAO/LAE: Left atrial overload/enlargement with bifid P-waves

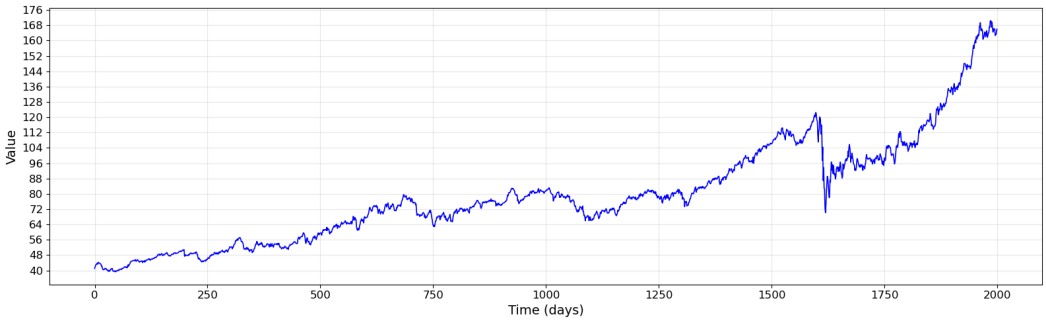

ECG 12
V1
V2
V3
V4
V5
V6
0.0 0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0 5.5 6.0 6.5 7.0 7.5 8.0 8.5 9.0 9.5
(s)

Finance Question Example

Q: Based on the daily closing price data for MAA over the past 2000 trading days, what does the Relative Strength Index (RSI) analysis reveal about the stock's momentum condition at the end of the period?

A. The stock is in overbought territory with RSI above 70, suggesting potential selling pressure.
B. The stock is in oversold territory with RSI below 30, suggesting potential buying opportunity.
C. The stock shows neutral momentum with RSI around 50, indicating balanced buying and selling pressure.
D. The stock shows strong upward momentum with RSI consistently increasing but not yet overbought.

answer: The stock shows neutral momentum with RSI around 50, indicating balanced buying and selling pressure.



D TIMESERIESEXAMAGENT LLM VERIFIER

For each template, we use an LLM to evaluate the generated question. Specifically, we ask:

- Is the question relevant to the given concept?
- Does answering the question require the provided time series?
- Are the question and answer free from ambiguity and bias?

D.1 VALIDATION PROMPT

```
You are an expert validator of question templates involving reasoning over
{exam_type} time series data.
You are given an exam question template:

{exam_template}

Your task is to validate the question template using the following criteria:
1. Is the question relevant to {exam_type} time series analysis?
2. Would you need the time series itself to answer the question?
3. Are there no ambiguity in the question or its answer?

If the answer to all is YES or MOSTLY YES, return only the number 1.
If the answer to either is NO, return your objections.
Return 1 (do not include any additional text then) or describe your objections.
```

E TIMESERIESEXAMAGENT OTHER DESIGN SPECIFICS

Detractors In addition, the mechanism of plausible but incorrect answer choices was implemented. The LLM is prompted to reflect on possible mistakes that the test taker might make while solving the exam. Using this knowledge, misleading, incorrect option choices can be generated.

Context Condensation A common issue we encountered in the framework was context window overflow during exam regeneration. To mitigate this, we applied context condensation, which reduces the number of tokens while preserving essential information. In our setup, the agent generates templates in a conversational manner. The process begins with a generation prompt, followed by a message containing the generated exam. If errors occur or the exam is rejected during verification, the feedback and regenerated exams are appended to the conversation. Several context condensation techniques exist, such as windowing (3) and context compression (30). We adopt a summarization-based method (39; 41), which has shown strong results in prior work and fits our use case. Specifically, we summarize non-recent pairs of failing exams and error messages into short descriptions that highlight the issues encountered. These summaries provide the LLM with concise feedback, supporting the generation of higher-quality templates.

RAG/web search In our setup, LLMs can also make use of external knowledge sources. The agent has two options: (i) a Retrieval-Augmented Generation (RAG) tool (22),, which pulls information from a structured corpus such as a PDF with domain materials, and (ii) web search, which provides access to more up-to-date or niche information. The retrieved content is then used to support concept generation, helping the model produce more accurate and comprehensive outputs.

F TIMESERIESEXAMAGENT AGENTIC FRAMEWORK HYPERPARAMETERS

In this section, we list all the hyperparameter used for our agentic workflow.

1. Generator LLM: the LLM used to generate concepts and the corresponding template. We used claude-sonnet-4-20250514 (initial generation with `reasoning_effort="medium"`). As a result, models developed by Anthropic are excluded from subsequent evaluations.
2. Concept LLM: the LLM used to generate concepts. We used gpt-4o-2024-08-06.
3. Verifier LLM: the LLM used to verify templates. We used gpt-4o-2024-08-06.
4. Student LLMs: the student LLMs we use to check the exam differentiability. Currently we have two student LLMs: stronger: gpt-4o-2024-08-06 and weaker: gpt-4o-mini. For each template under evaluation, students receive the same set of 3 samples to answer.
5. Exam type: We are generating the data connected to specific domain. We used "ecg", "medicine", "finance", "weather" and "mechanical".
6. Few-shot examples: 9 templates prepared beforehand were used to present the desired structure to the generator LLM. For each generation, 3 templates were randomly selected and included in the prompt as few-shot examples. This introduces variability into the generation process, enhancing diversity.
7. Regeneration patience: Templates requiring multiple regeneration cycles were generally of lower quality. In our experiments, we set a maximum of 3 regeneration attempts.

G FINE-GRINED EVALUATION OF GENERATED RESULT FROM TIMESERIESEXAMAGENT

G.1 LLM-AS-A-JURY

We evaluated a set of generated questions using the LLM-as-a-jury approach. Below are example criteria we applied for ECG evaluation:

```
1. SPECIFICITY
You are an expert judge evaluating the specificity of ECG multiple
   -choice questions.
The questions normally come together with relevant time series
   data, which should be analized to answer the question
   correctly. It is not includded in currently evaluated samples.

Evaluate the specificity of the generated ECG multiple-choice
   question.

A good question should target a single phenomenon.

Evaluation steps:
1. Read the question and all answer options.
2. Determine if the question targets a single, clearly defined ECG
    finding or clinical interpretation.
3. Assess the ratio of unique medical terms to general words.
4. Penalize if:
   - The question is overly broad or open-ended (e.g., "Is this
   ECG normal?").
   - The wording leaves diagnostic interpretation unclear.
   - The question covers multiple unrelated phenomena.

Score highest if the question has one precise focus (e.g., "Is
   there ST elevation in lead V3?").

Score from 1-10 where:
- 10: Excellent specificity with clear, focused medical
   terminology targeting a single phenomenon
- 7-9: Good specificity but could be more focused
- 4-6: Moderate specificity with some clarity issues
- 1-3: Poor specificity, too broad, or covers multiple unrelated
   phenomena

Respond with just a number from 1 to 10, followed by a brief
   explanation for your score.

2. UNAMBIGUITY
You are an expert judge evaluating the unambiguity of ECG multiple
   -choice questions.
The questions normally come together with relevant time series
   data, which should be analized to answer the question
   correctly. It is not included in currently evaluated samples.
Task: Evaluate if the question and answers can be objectively
   assessed without multiple interpretations.

Evaluation steps:
1. Read the question and all answer options.
2. Determine if the question can be objectively assessed.
3. Check if the answers are clear and unambiguous.
4. Penalize if:
```

```
   - The question uses subjective terms (e.g., "Does this look
   strange?").
   - The answers are open to multiple interpretations.
   - The question cannot be objectively answered.


A good question should be clear and objective (e.g., "Is there
   tachycardia?").

Score from 1-10 where:
- 10: Completely unambiguous and objective with crystal clear
   question and answers
- 7-9: Mostly clear with only minor ambiguities
- 4-6: Moderately clear but has some ambiguous elements
- 1-3: Highly ambiguous, subjective, or open to multiple
   interpretations

Respond with just a number from 1 to 10, followed by a brief
   explanation for your score.

3. DOMAIN RELEVANCE
You are an expert judge evaluating the domain relevance of ECG
   multiple-choice questions.
The questions normally come together with relevant time series
   data, which should be analized to answer the question
   correctly. It is not includded in currently evaluated samples.
Task: Evaluate if the question actually pertains to ECGs and
   medicine.

Evaluation criteria:
1. Does the question contain medical and ECG-specific terminology?
2. Is the question relevant to ECG interpretation and medical
   diagnosis?
3. Is the question related to ECG interpretation?
4. Does the question have proper medical context?

A good question should contain relevant medical terms (e.g., "QRS
   ," "arrhythmia," "P wave") and pertain to ECG interpretation.

Score from 1-10 where:
- 10: Highly relevant to ECG domain with extensive proper medical
   terminology
- 7-9: Good domain relevance with appropriate medical terms
- 4-6: Moderate relevance but could be more medically specific
- 1-3: Poor medical relevance or contains primarily non-medical
   terms

Respond with just a number from 1 to 10, followed by a brief
   explanation for your score.

3. ANSWERABILITY
You are an expert judge evaluating the answerability of ECG
   multiple-choice questions.
The questions normally come together with relevant time series
   data, which should be analized to answer the question
   correctly. It is not includded in currently evaluated samples.
Task: Evaluate if the question can be answered based on ECG data
   analysis.
```

```
Evaluation steps:
1. Read the question and all answer options.
2. Determine if the question can be answered by analyzing ECG
    waveform data.
3. Assess whether the question requires time series analysis or
    could be answered without it.
4. Penalize if:
    - The question asks about non-ECG factors (e.g., "Was the
    patient nervous?").
    - The question can be answered without analyzing the ECG time
    series data.
    - The question is too general and doesn't require specific ECG
    analysis.

Score highest if the question requires specific ECG time series
    analysis (e.g., "Is there atrial fibrillation?").
Give fewer points if the question can be answered without time
    series data.

Score from 1-10 where:
- 10: Requires specific, detailed ECG analysis and is fully
    answerable from the data
- 7-9: Mostly answerable from ECG data but could be more specific
- 4-6: Partially answerable from ECG but has some limitations
- 1-3: Cannot be answered from ECG data or is too general/
    unrelated

Respond with just a number from 1 to 10, followed by a brief
    explanation for your score.
```

G.2 t-SNE Embedding Plots

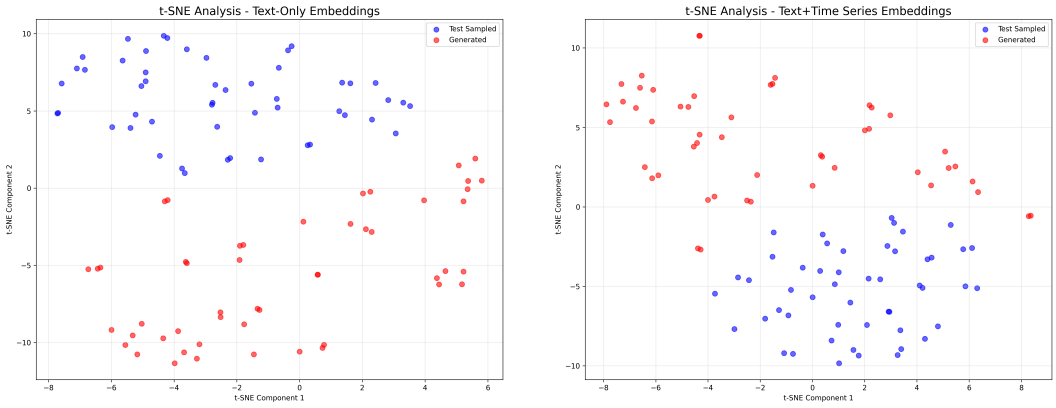


Figure 6: t-SNE analysis of embeddings: (left) text-only vs. (right) text and time series concatenated together.

To visualize distributional differences, we applied t-SNE (26), which preserves local distances between samples. As shown in Fig. 6, questions generated by our framework form a more widely scattered distribution, confirming the higher diversity observed in Table 5.

We ran the t-SNE algorithm using the scikit-learn implementation with the random seed fixed to 42, in order to ensure full reproducibility of the dimensionality reduction results across different runs. The other parameters were set to default.

The text embeddings were generated using the SentenceTransformer model (Qwen3-Embedding-8B), while the time-series embeddings were obtained from MOMENT-1-large and averaged across leads or multiple time series when applicable. The two vectors were then combined through direct concatenation to form a joint embedding.

H EVALUATION PROTOCOL

All used models were accessed by API with `LiteLLM` Python library. The following API providers were used with default parameters:

- Closed source models – OpenAI API, Anthropic API
- Open source models for `TimeSeriesExamAgent` generated exams – Hugging Face Inference Providers API
- Open source models for `TimeSeriesExam`– Novita AI [3]

During the evaluation, the images of the plots were encoded with base64 encoding and provided to the models. Plots were created with DPI = 50. We used setup without context condensation.

[3] `https://novita.ai/`

I FINETUNE PARAMETERS

Hyperparameter	**Value**
Base model	Qwen2.5-VL-3B-Instruct
GPU setupe	4*NVIDIA RTX A6000 48GB GPU
Frameworks	Hugging Face Accelerate, DeepSpeed ZeRO 3 stage
Train samples	2000
Warm-up steps	16
Batch size per device	1
Gradient accumulation steps	8
Learning rate	5e-5
Optimizer	AdamW
Learning rate scheduler	Cosine
Weight decay	0.1
LoRA rank (r)	16
LoRA alpha	16
LoRA dropout	0.0
LoRA target modules	q_proj, k_proj, v_proj, o_proj, gate_proj, up_proj, down_proj

J FEEDBACK IMPACT

One of the challenges we observed during question generation was the misuse of domain-specific jargon. Although the generated questions were grammatically correct, they sometimes included terminology that did not align with standard ECG practice. This can lead to confusion for clinicians, as non-standard phrasing undermines clarity and clinical relevance.

The following generated question contains terminology that was later identified as suboptimal:

> **Q:** Examine this Lead II ECG recording and measure the QRS voltage amplitudes throughout the tracing. Based on the peak-to-peak QRS amplitudes observed, what voltage abnormality is present?

The clinicians noted that certain expressions in the question do not reflect standard ECG terminology. In particular, the phrase *"peak-to-peak QRS"* was considered inappropriate. To address this, the natural language description was refined by adding the following instruction:

```
Please frame your questions in a way that is clear and natural for
    ECG specialists (i.e., adjust terminology accordingly).
```

Following this modification, a second round of consultation confirmed that the issue of non-standard jargon had been resolved. An example of an improved question generated with the revised prompt is shown below:

> **Q:** Based on QRS voltage amplitude measurements across all 12 leads in this ECG, which ventricular condition is most likely present?

K OTHER DATASETS

To further assess the generalization capabilities of our solution, we generate exams from the bearing domain (e.g., the Case Western Reserve University Bearing Dataset) with the following prompt:

```
I want to create time series exam testing model understanding of
   bearing vibration signals for fault diagnosis.

To load the data, use the provided user_dataset object. Use its
   method query: def query(self, sample_id: int) -> np.ndarray:

The CWRU (Case Western Reserve University) Bearing Dataset is a
   widely-used dataset for bearing fault diagnosis. It contains
   vibration signals from bearings under different fault
   conditions. The dataset includes 4600 samples of 32x32 time
   series data representing bearing vibration measurements at 48
   kHz sampling rate.

The dataset contains 10 different bearing conditions:
- Normal: Healthy bearing operation
- Ball_007, Ball_014, Ball_021: Ball bearing faults with different
    severities (0.007", 0.014", 0.021" diameter)
- IR_007, IR_014, IR_021: Inner race faults with different
   severities
- OR_007, OR_014, OR_021: Outer race faults with different
   severities

You can focus some of the questions around these fault types and
   characteristics:

FAULT TYPES:
```

```
- Normal: Normal bearing operation
- Ball: Ball bearing element fault
- IR: Inner race fault
- OR: Outer race fault

SEVERITY LEVELS:
- 007: Mild fault (0.007 inch diameter)
- 014: Moderate fault (0.014 inch diameter)
- 021: Severe fault (0.021 inch diameter)

FAULT CHARACTERISTICS:
- Ball faults typically show periodic impacts at ball pass
    frequency
- Inner race faults show impacts at inner race frequency
- Outer race faults show impacts at outer race frequency
- Normal bearings show random vibration patterns
- Fault severity affects signal amplitude and frequency content
- Different fault locations create distinct frequency signatures

Make sure that the question cannot be answered without the
    timeseries - it is very important. We don't want LLMs to be
    able to guess the answer easily.

Template should return the time series (one or two of them) in
    same form as was received from '''query''' -- do not flatten
    it!
```

L LLM USAGE STATEMENT

We used Large Language Models (LLMs) only as a writing aid, for polishing the text and improving clarity and grammar.