Healthcare TimeSeries Reasoning Benchmarks at Scale

Anonymous Author(s)
Affiliation
Address
email

Abstract

We introduce `TimeSeriesExamAgent`, a scalable and domain-agnostic framework for automatically generating and validating time series reasoning benchmarks. Existing benchmarks lack scalability, are limited to a few specific domains, while building them remains labor intensive. Automated solutions for benchmark creation have been proposed, but these typically rely on a single-step generation process without verification, leading to lower-quality exams. Our framework addresses these limitations by enabling stakeholders—such as institutions with highly confidential data—to easily construct high-quality, domain-specific benchmarks from their own private datasets. A domain expert provides a dataset, a natural language description, and a simple data-loading method. The agent then orchestrates the generation pipeline, including creating question templates, robustness verification from multiple perspectives, and iterative refinement. We demonstrate the framework on two medical datasets and evaluate multiple state-of-the-art language models on the generated benchmarks. Empirically, we demonstrate that the framework produces domain-agnostic benchmarks whose diversity matches human-generated counterparts, and our evaluation of several Large Language Models shows that accuracy remains limited, underscoring open challenges in time-series reasoning.

1 Introduction

Many recent works have applied Large Language Models (LLMs) to time series analysis tasks such as forecasting, anomaly detection, and classification [1, 2, 3, 4, 5, 6]. More recently, attention has shifted to evaluating the reasoning capabilities of LLMs in time series tasks. These evaluations are typically framed in two ways: 1) contextualized traditional tasks such as forecasting, but with added contextual information (e.g., providing a clinical scenario before a prediction) [7, 8, 9, 10, 11], and 2) reasoning and understanding tasks that directly probe concepts in time series (e.g., "what kind of trend does the following series exhibit?") [12, 13].

However, existing benchmarks have clear limitations. Contextualized tasks remain close to traditional metrics (e.g., mean-squared-error for forecasting) without testing deeper reasoning, while reasoning-style benchmarks often focus only on simple properties like trend or seasonality. In practice, real-world domains such as healthcare require more complex reasoning, where tasks like diagnosis naturally combine anomaly detection, classification, and domain knowledge. Curation is another challenge. Annotation or template-based benchmarks are labor-intensive, while LLM-based augmentation often lacks diversity because it simply expands existing datasets. As a result, building specialized, domain-specific benchmarks remains difficult and time-consuming.

This challenge is especially pronounced in healthcare. Clinical datasets are both highly sensitive and highly specialized: tasks in cardiology, radiology, or genomics often require nuanced reasoning that combines statistical patterns with medical expertise and domain knowledge. Unlike open domains

Submitted to 39th Conference on Neural Information Processing Systems (NeurIPS 2025). Do not distribute.

where public benchmarks are available, stakeholders in healthcare cannot easily share their datasets due to patient privacy, regulatory constraints, and ethical concerns. At the same time, they need ways to rigorously evaluate whether generative models can reason about ECG signals, imaging studies, or longitudinal patient records without exposing protected health information. This creates an urgent need for customizable, automated benchmark generation tailored to private clinical datasets.

Inspired by recent agent-based approaches in other domains [14, 15], we propose `TimeSeriesExamAgent`, a pipeline that (1) generates domain-specific multiple-choice questions on time-series data, (2) scales efficiently, and (3) ensures reliable ground truth through iterative verification. We also evaluate four state-of-the-art LLMs on a benchmark of 348 automatically generated questions samples. Our results show that many models struggle on highly complex tasks that require combining quantitative analysis with domain knowledge. For brevity, we provide detailed related work in Appendix A.

2 TimeSeriesExamAgent

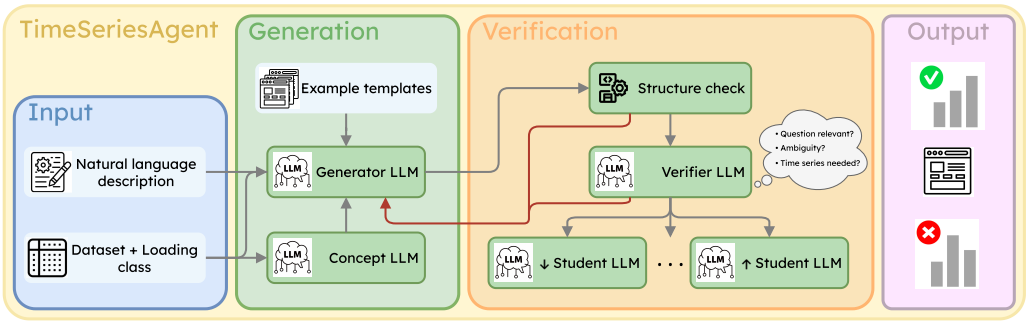


Figure 1: TimeSeriesExamAgent architecture. The user provides exam-making instructions and a custom dataset with minimal loading code. Agent outputs question templates – Python functions generated by a generator LLM and filtered through three progressive stages of verification (syntax and output format check, validation by LLM judge, capability-aligned filtering). Arrows denote data flow, red ones show direction for rejected templates.

In this section, we introduce `TimeSeriesExamAgent`, a multi-agent framework that combines planning, generation, and verification to enable automatic benchmark construction. In this section, we describe `TimeSeriesExamAgent` and its workflow in detail. An overview is shown in Fig. 1. 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 We generate question templates instead of samples directly, as shown in Fig. 2. 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 Appendix B. 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.

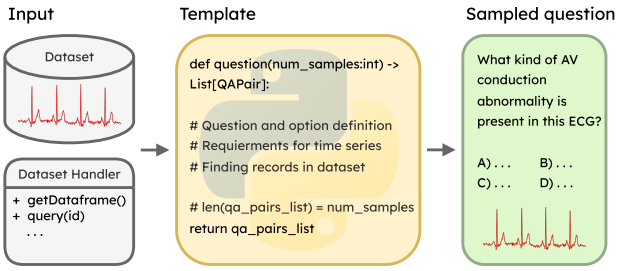


Figure 2: Question generation process: With information about dataset, TimeSeriesAgent generates question template in a form of Python functions. The created function can be called to get arbitrary number of question samples.

Structure verification We check whether the generated template can be executed successfully. We execute the generated template $k = 5$ times; if there are failures, the error message is returned as a feedback.

Content verification Certain aspects of quality control are particularly well-suited for 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 validty of each template. A quantitative score is given, and we set a threshold for rejection. If the verifier raises any rejection, its explanation is treated similarly to a structural error and the template is regenerated. We provide the detailed prompt in Appendix C.

Capability-Aligned Filtering 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 inspired by educational theory, particularly the expertise reversal effect [16]. 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 hyper-parameters in Appendix F and other design specifics in Appendix D

3 Experimental Setup, Results and Discussion

First, we generate one exam for each of the two real world datasets: PTB-XL [17], MIT-BIH [18]. In total, we have 197 samples for MIT-BIH, and 151 samples for PTB-XL. 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.

We select candidate models to cover a diverse range of performance levels, as indicated by the OpenVLM Leaderboard [23]. In both cases, the best results achieves Gemma-3-27b-it, which outperforms the remaining models.

To further evaluate our benchmark, we compare multiple metrics on questions generated from the dataset with those in `ECG-QA` [10], a template-based benchmark also built on `PTB-XL`. The goal is to demonstrate that our framework achieves comparable diversity without requiring manual template curation. We picked random 50 question samples from each benchmark and calculated the distances for every possible pair within the set. We used the `Qwen/Qwen3-0.6B` sentence transformer model to extract embeddings, as it achieved the second-best performance among all models on the Hugging Face MTEB leaderboard.

	Dataset	
Model	**MIT-BIH**	**PTB-XL**
gpt-4o [19]	0.416	0.424
o3-mini [20]	0.442	0.477
Qwen2.5-VL-Instruct [21]	0.411	0.490
Gemma-3-27b-it [22]	**0.497**	**0.517**

Table 1: Comparative performance of four vision-language models across medical (MIT-BIH, PTB-XL) time-series datasets.

	Mean ± Std	
Benchmark Dataset	Embedding	Normalized Levenshtein
ECG-QA	0.207 ± 0.079	0.519 ± 0.157
TimeSeriesExamAgent (ours)	$\mathbf{0.301 \pm 0.070}$	$\mathbf{0.542 \pm 0.039}$

Table 2: Question diversity comparison using embedding and normalized Levenshtein distance.

As shown in Table 2, benchmark generated by our framework shows a diversity comparable to one developed by humans. This indicates that the proposed framework is able to capture a wide range of expressions without relying on handcrafted templates, supporting its scalability and adaptability to other domains. We also employed G-Eval, a probabilistic LLM-as-a-judge framework [24]. An LLM is used to evaluate the relevance of each question, assigning a score between 0 and 1 to indicate how well it meets the specified criteria. Results are presented in Table 3. We provide the detailed G-Eval prompt in Appendix E.

	Mean Result			
Dataset	Specificity	Unambiguity	Domain Relevance	Answerability
ECG-QA	0.604	0.562	0.827	0.898
TimeSeriesExamAgent (ours)	**0.922**	**0.932**	**0.989**	**0.992**

Table 3: Question diversity comparison using G-Eval framework.

4 Limitations and Conclusions

In this work, we present a scalable, domain-specific framework for the automatic generation of time-series benchmarks, enabling the creation of high-quality, large-scale evaluation datasets while minimizing the need for labor-intensive human annotation. A limitation of this study is that the quality of the generated exams depends on the quality and coverage of the time series dataset. Additionally, domain specialists must provide carefully crafted prompts.

For future work, we will explore human-in-the-loop improvements to template generation. In offline sessions with clinicians, we observed that exams produced with such feedback are more likely to be deemed valid. We also plan to validate exam quality by training time series–text alignment models and testing their transfer performance on other established reasoning benchmarks [8]. Finally, there is growing attention on building time series agentic frameworks [25, 26]. Enabling these frameworks to write code in order to answer our benchmark questions would provide valuable insights to the community.

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] 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.

[3] 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.

[4] 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.

[5] 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.

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

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

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

[13] Willa Potosnak, Cristian Challu, Mononito Goswami, Kin G. Olivares, Michał Wiliński, Nina Żukowska, and Artur Dubrawski. Investigating compositional reasoning in time series foundation models, 2025.

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

[15] 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.

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

[17] 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.

[18] 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.

[19] 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.

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

[21] 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.

[22] 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.

[23] 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.

[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] 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.

[26] 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.

[27] 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.

[28] 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.

[29] 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.

[30] 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.

[31] 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.

[32] 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.

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

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

[35] 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.

[36] 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.

A Related Work

Time series benchmarks The task of creating domain-specific time series reasoning benchmarks is challenging. Existing benchmarks are either domain-agnostic, or limited to a specific domains with high quality datasets. For example, `TimeSeriesExam` [12] introduced over 700 multiple-choice questions to evaluate five general reasoning skills, but its questions primarily assess signal properties (e.g. trend, cyclicity, stationarity) and lack the contextual depth needed for real-world applications. Domain-specific benchmarks address this gap but have limited scope and poor extensibility, since their curation often relies on templates. For instance, ECG-QA [10] and ECG-Expert-QA [11] focus on ECG interpretation, while EngineMT-QA [27] 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 [8, 9], which is both difficult and time-consuming—undermining the main advantage of automation.

Title	Multi-Domain	Curation	# Samples	Skill type		
		Fully Automatic		P	R	PS
Time-MQA [8]	✓	✗	200,000	✓	✓	✓
TimeSeriesExam [12]	✗	✗	763	✓	✓	✗
Time-MMD [9]	✓	✗	17,113	✓	✗	✗
MT-Bench [7]	✓	✓	22,000	✓	✓	✗
ECG-QA [10]	✗	✗	414,348	✓	✓	✗
TimeSeriesExamAgent (ours)	✓	✓	600+	✓	✓	✓

Table 4: Overview of time series and multimodal datasets with curation and skill types (P – Prediction, R – Reasoning, PS – Practical skills (tasks beyond classification and reasoning such as performing calculations and applying formulas). TimeSeriesExamAgent is universal – tailored to user's needs and with advanced automatic verifications.

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. In LLM-based settings, the language model often provides the reasoning or planning layer that guides the agent's decisions. 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 [14]. For instance, [28] integrates exploratory evaluation using reinforcement learning, while [14] 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 [29].

B Generation Agent Workflow

We rely on two stages of generation for the templates: planning and generating, inspired by the chain-of-thought (CoT) prompting[30].

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[31].

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 [32].

B.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.
```

B.2 Example of Question Template

```
def question_hypertrophic_cardiomyopathy(num_samples, verbose=False):
    hyperparameters = {
        "min_probability_threshold": 75.0,
        "target_abnormalities": ["SEHYP", "LVH", "VCLVH", "RVH"],
        "normal_codes": ["NORM"],
        "max_attempts": 1000,
    }

    question = "Based on the morphological characteristics and voltage
    patterns observed in this ECG recording, what is the most likely
    structural cardiac finding?"

    options = [
```

```
        "Septal hypertrophy with voltage criteria consistent with
    hypertrophic cardiomyopathy",
        "Left ventricular hypertrophy with strain pattern indicating
    pressure overload",
        "Right ventricular hypertrophy suggesting pulmonary hypertension",
        "Normal cardiac structure with physiological variant morphology",
    ]

    def has_high_probability_abnormality(scp_codes_dict, target_codes,
    threshold):
        for code in target_codes:
            if code in scp_codes_dict and scp_codes_dict[code] >= threshold:

                return code
        return None

    def is_normal_ecg(scp_codes_dict, normal_codes, threshold):
        for code in normal_codes:
            if code in scp_codes_dict and scp_codes_dict[code] >= threshold:

                return True
        return False

    qa_pairs = []
    attempts = 0
    df = user_dataset.get_dataframe()

    while len(qa_pairs) < num_samples and attempts < hyperparameters["
    max_attempts"]:
        attempts += 1
        if verbose:
            print(f"[Hypertrophic Cardiomyopathy] Generating question {len(
    qa_pairs)+1}/{num_samples}")

        # Sample a random record
        sample_row = df.sample(1).iloc[0]
        ecg_id = sample_row['ecg_id']
        scp_codes = sample_row['scp_codes']

        if not isinstance(scp_codes, dict):
            continue

        # Check for target abnormalities with high probability
        detected_abnormality = has_high_probability_abnormality(
            scp_codes, hyperparameters["target_abnormalities"],
    hyperparameters["min_probability_threshold"]
        )

        is_normal = is_normal_ecg(scp_codes, hyperparameters["normal_codes
    "], hyperparameters["min_probability_threshold"])

        if detected_abnormality is None and not is_normal:
            continue

        try:
            ts = user_dataset.query(ecg_id)
            if ts is None or ts.shape != (12, 1000):
                continue

```

```
        except Exception as e:
            if verbose:
                print(f"Error loading ECG {ecg_id}: {e}")
            continue

        # Determine correct answer based on detected abnormality
        if detected_abnormality == "SEHYP":
            correct_answer = options[0]
            answer_type = "septal_hypertrophy"
        elif detected_abnormality == "LVH" or detected_abnormality == "
    VCLVH":
            correct_answer = options[1]
            answer_type = "left_ventricular_hypertrophy"
        elif detected_abnormality == "RVH":
            correct_answer = options[2]
            answer_type = "right_ventricular_hypertrophy"
        elif is_normal:
            correct_answer = options[3]
            answer_type = "normal"
        else:
            continue

        qa_pairs.append({
            "question": question,
            "options": options,
            "answer": correct_answer,
            "answer_type": answer_type,
            "ecg_id": ecg_id,
            "ts": ts.tolist(),
            "scp_codes": scp_codes,
            "detected_abnormality": detected_abnormality,
            "relevant_concepts": ["Hypertrophic Cardiomyopathy", "
    Structural Heart Disease", "ECG Morphology", "Voltage Criteria"],
            "domain": "cardiology",
            "detractor_types": ["Similar structural abnormalities", "Normal
     variants"],
            "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 needed to answer the question
    is given. If you are unsure, please provide your best guess.",
        })

    return qa_pairs
```

B.3 Example of Natural Language Description

```
I want to create time series exam testing model understanding of ecg
    signals.

To load the data, use the provided user_dataset object.

The PTB-XL ECG dataset is a large dataset of 21799 clinical 12-lead ECGs
    from 18869 patients
of 10 second length. The raw waveform data was annotated by up to two
    cardiologists, who assigned
potentially multiple ECG statements to each record. The in total 71
    different ECG statements conform
```

to the SCP-ECG standard and cover diagnostic, form, and rhythm statements.
    The dataset is complemented by extensive metadata on demographics,
    infarction characteristics, likelihoods for diagnostic ECG statements
    as well as annotated signal properties.

You can focus some of the questions around those scp codes:

NDT	non-diagnostic T abnormalities
NST_	non-specific ST changes
DIG	digitalis-effect
LNGQT	long QT-interval
NORM	normal ECG
IMI	inferior myocardial infarction
ASMI	anteroseptal myocardial infarction
LVH	left ventricular hypertrophy
LAFB	left anterior fascicular block
ISC_	non-specific ischemic
IRBBB	incomplete right bundle branch block
1AVB	first degree AV block
IVCD	non-specific intraventricular conduction disturbance (block)
ISCAL	ischemic in anterolateral leads
CRBBB	complete right bundle branch block
CLBBB	complete left bundle branch block
ILMI	inferolateral myocardial infarction
LAO/LAE	left atrial overload/enlargement
AMI	anterior myocardial infarction
ALMI	anterolateral myocardial infarction
ISCIN	ischemic in inferior leads
INJAS	subendocardial injury in anteroseptal leads
LMI	lateral myocardial infarction
ISCIL	ischemic in inferolateral leads
LPFB	left posterior fascicular block
ISCAS	ischemic in anteroseptal leads
INJAL	subendocardial injury in anterolateral leads
ISCLA	ischemic in lateral leads
RVH	right ventricular hypertrophy
ANEUR	ST-T changes compatible with ventricular aneurysm
RAO/RAE	right atrial overload/enlargement
EL	electrolytic disturbance or drug (former EDIS)
WPW	Wolf-Parkinson-White syndrome
ILBBB	incomplete left bundle branch block
IPLMI	inferoposterolateral myocardial infarction
ISCAN	ischemic in anterior leads
IPMI	inferoposterior myocardial infarction
SEHYP	septal hypertrophy
INJIN	subendocardial injury in inferior leads
INJLA	subendocardial injury in lateral leads
PMI	posterior myocardial infarction
3AVB	third degree AV block
INJIL	subendocardial injury in inferolateral leads
2AVB	second degree AV block
ABQRS	abnormal QRS
PVC	ventricular premature complex
STD_	non-specific ST depression
VCLVH	voltage criteria (QRS) for left ventricular hypertrophy
QWAVE	Q waves present
LOWT	low amplitude T-waves
NT_	non-specific T-wave changes
PAC	atrial premature complex

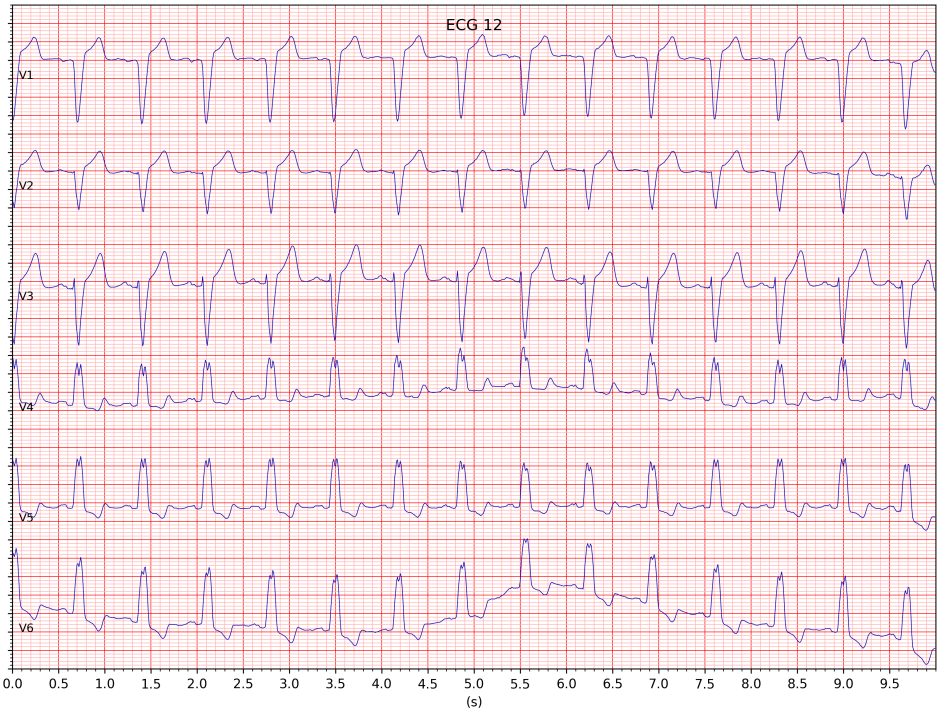
```
LPR     prolonged PR interval
INVT    inverted T-waves
LVOLT   low QRS voltages in the frontal and horizontal leads
HVOLT   high QRS voltage
TAB_    T-wave abnormality
STE_    non-specific ST elevation
PRC(S)  premature complex(es)
SR      sinus rhythm
AFIB    atrial fibrillation
STACH   sinus tachycardia
SARRH   sinus arrhythmia
SBRAD   sinus bradycardia
PACE    normal functioning artificial pacemaker
SVARR   supraventricular arrhythmia
BIGU    bigeminal pattern (unknown origin, SV or Ventricular)
AFLT    atrial flutter
SVTAC   supraventricular tachycardia
PSVT    paroxysmal supraventricular tachycardia
TRIGU   trigeminal pattern (unknown origin, SV or Ventricular)
```

B.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


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)

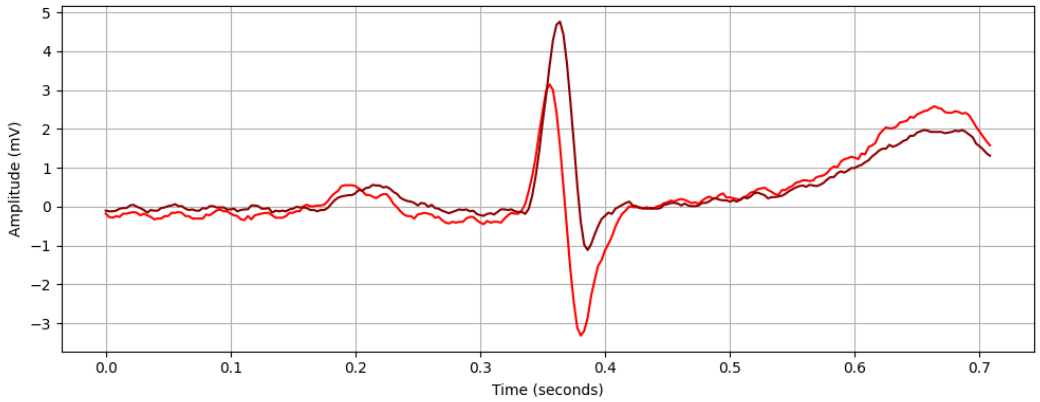
Q: Examine the ECG at shown. Which of the following statements best describes the T-wave morphology in this single-beat ECG?

A. The T-wave is upright (positive), suggesting normal ventricular repolarization.

B. The T-wave is inverted (negative), which may indicate myocardial ischemia or ventricular strain.

C. The T-wave is biphasic (partly positive, partly negative), which may indicate regional repolarization abnormalities.

D. The T-wave is flattened, which may indicate electrolyte disturbances such as hypokalemia.



C 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?

C.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.
```

D 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 [33] and context compression [34]. We adopt a summarization-based method [35, 36], 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.

E G-Eval

We evaluated a set of generated questions under the G-Eval framework. We used the following criteria:

```
1. SPECIFICITY
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?").

1. UNAMBIGUITY
Evaluate the unambiguity of the generated ECG multiple-choice question.
A question and the answers should not have 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.

Score highest if the question is clear and objective (e.g., "Is there
    tachycardia?"),

2. DOMAIN RELEVANCE
```

```
Evaluate the domain relevance of the generated ECG multiple-choice question.

Does the question actually pertain to ECGs and medicine?

Evaluation steps:
1. Read the question and all answer options.
2. Identify medical and ECG-specific terminology.
3. Determine if the question is relevant to ECG interpretation and medical
    diagnosis.
4. Penalize if:
   - The question contains non-medical terms (e.g., "Is the line pretty?").
   - The question is not related to ECG interpretation.
   - The question lacks medical context.

Score highest if the question contains relevant medical terms
(e.g., "QRS," "arrhythmia," "P wave") and pertains to ECG interpretation.

3. ANSWERABILITY
Evaluate the answerability of the generated ECG multiple-choice question.
Even without an answer provided, the question should be answerable based on
     the data (ECG).

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.
```

F Hyperparameters

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

1. Generator LLM: the LLM use to generate concepts and the corresponding template. We used claude-sonnet-4-20250514 (initial generation with `reasoning_effort="medium"`).
2. Concept LLM: the LLM use to generate concepts. We used gpt-4o-2024-08-06.
3. Verifier LLM: the LLM use 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.
5. Exam type: We are generating the data connected to specific domain. We used "ECG".
6. Few-shot examples: 3 templates prepared beforehand were used to present the desired structure. For each generation, they were randomly sampled from set of 9.

G 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 – Hugging Face Inference Providers API

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.

H Expert evaluation

We also presented samples of our work to the clinicians. During the first meeting, we received feedback that our work was interesting but required further improvements. Specifically, the plots needed to be both stretched, annotated and all 12 leads needed to be included. In addition, the language and jargon we used could be confusing for specialists. Overall, the clinicians considered 3 out of 5 questions to be answerable and flagged 2 out of 5 as problematic.

At the following meeting, after incorporating experts' comments into the prompt, we asked a specialist to provide their opinion on improvements on the exams, using the following criteria:

- **Correctness:** The answer must be unequivocally accurate according to current medical knowledge and guidelines.
- **No Ambiguity:** Only one answer should be valid; distractors must be plausible but clearly incorrect.
- **Precision of Wording:** Both questions and answers should be clear, concise, and medically accurate, avoiding vague phrasing.
- **Relevance:** The question should be engaging and meaningful to the specialist.

Of the 8 questions evaluated, 7 were rated positively across all four criteria: correctness, lack of ambiguity, precision of wording, and relevance.