

ARFBENCH: BENCHMARKING MULTIMODAL TIME SERIES REASONING FOR SOFTWARE INCIDENT RESPONSE

Stephan Xie^{1,2}, Ben Cohen², Mononito Goswami³, Junhong Shen¹,
Emaad Khwaja², Chenghao Liu², David Asker², Othmane Abou-Amal², Ameet Talwalkar^{1,2}

¹Machine Learning Department, Carnegie Mellon University, Pittsburgh, PA, USA

²Datadog AI Research, New York, NY, USA

³Amazon AI Research, Seattle, WA, USA

stephan.xie@datadoghq.com, ameer.talwalkar@datadoghq.com

ABSTRACT

Time series question-answering (TSQA), in which we ask natural language questions to infer and reason about properties of time series, is a promising yet under-explored capability of foundation models. In this work, we present ARFBENCH, a TSQA benchmark that evaluates the understanding of multimodal foundation models (FMs) on time series anomalies prevalent in software incident data. ARFBENCH consists of 750 questions across 142 time series and 5.38M data points from 63 production incidents sourced exclusively from internal telemetry at Datadog. We evaluate leading proprietary and open-source LLMs, VLMs, and time series FMs and observe that frontier VLMs perform markedly better than existing baselines; the leading model (GPT-5) achieves a 62.7% accuracy and 51.9% F1. We next demonstrate the promise of specialized multimodal approaches. We develop a novel TSFM + VLM hybrid prototype which we post-train on a small set of synthetic and real data that yields comparable overall F1 and accuracy with frontier models. Lastly, we find models and human domain experts exhibit complementary strengths. We define a model-expert oracle, a best-of-2 oracle selector over model and expert answers, yielding 82.7% F1 and 87% accuracy and establishing a new superhuman frontier for future TSQA models. The benchmark is available at <https://huggingface.co/datasets/Datadog/ARFBench>.

Track: Research

1 INTRODUCTION

Trillions of dollars are lost each year due to system outages (Krasner, 2018). In order to resolve unexpected software incidents in a timely manner, engineers must decide on their next actions and determine where errors occur. To guide these choices, engineers rely on *observability metrics*, or time series used to monitor the health of production software such as CPU usage, latency, and error counts, among others. Engineers investigate incidents by repeatedly asking questions about these time series, such as “*Is the pattern in this time series abnormal?*”, or “*Are these two time series correlated?*”. These questions naturally give rise to time series question answering (TSQA) tasks, in which reasoning over multiple time series and temporal patterns is necessary in order to support downstream decision-making. Despite the central role of such question-driven analysis in incident response, it remains unclear whether modern foundation models can reliably answer the kinds of time series questions engineers ask in practice.

To this end, we present the Anomaly Reasoning Framework BENCHMARK (ARFBENCH), a multimodal, multiple-choice question-answering benchmark designed to test time series reasoning and understanding tasks that are constructed from real software incidents sourced exclusively from internal telemetry at Datadog, with sensitive data anonymized by both an LLM and manual review. Compared to existing benchmarks (Cai et al., 2024a; Kong et al., 2025; Wang et al., 2025b; Yang et al., 2025; Feng et al., 2025), ARFBench is distinguished by three key aspects: (i) it is grounded in real production time series from operational systems, (ii) each QA pair is supported by expert annotations and additional incident context, and (iii) tasks are designed to evaluate compositional

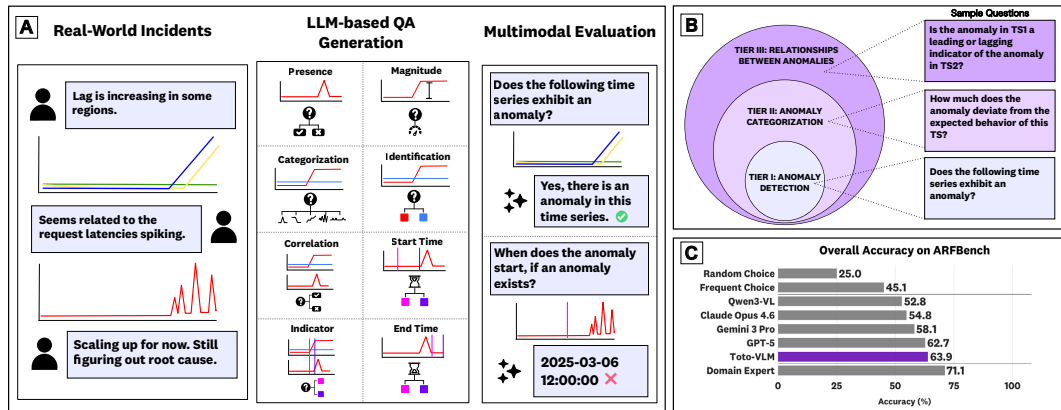


Figure 1: ARFBENCH consists of 750 question-answer (QA) pairs, derived from 63 real-world incidents and 142 observability time series. Observability time series are highly nonstationary and complex (Cohen et al., 2025), and ARFBENCH includes highly multivariate series that challenge LLM/VLM input representations. **A.** Workflow of ARFBench question-answer generation. Engineers use commercial messaging platforms to respond to incidents, where they typically send time series widgets that visualize relevant metrics. Time series and incident timelines from internally monitored incidents are used as input to an LLM pipeline and fit to eight different question templates testing various aspects of anomalies. The resulting multiple choice question-answer pairs can be used to evaluate various predictive models. **B.** ARFBench questions are designed to capture escalating levels of difficulty. **C.** Our model, Toto-VLM, obtains high accuracy and F1 on ARFBench, comparable to recent frontier models.

anomaly reasoning, with questions organized into three tiers of increasing difficulty, with higher-tier tasks depending on correct reasoning over lower tiers. A more extensive coverage of prior work is given in Appendix B.

ARFBench questions are designed in tiers that capture increasing levels of reasoning: first detecting anomalies, then identifying properties of anomalies, and lastly reasoning about the relationships between anomalies across time series. These questions test general time series understanding abilities such as event classification, magnitude estimation, trend and periodicity analysis, and also require robust understanding of correlation and temporal relationships (Section 2.1).

We benchmark a wide variety of foundation models on ARFBench, including leading proprietary and open-source LLMs, vision-language models (VLMs), and time series LLMs (Section D.2). However, we find that the high number of variates in ARFBench data poses major problems for plotting and context-length limits in both LLMs and VLMs, motivating a need for specialized time series modeling. Therefore, we also devise novel hybrid modeling methods by combining time series foundation models (TSFM) with VLMs.

We observe three key takeaways from our experiments.

1. **Frontier VLMs Markedly Lead Over Existing Baselines.** GPT-5, the top pre-existing model we evaluate, achieves 51.9% macro F1 and 62.7% accuracy, surpassing naive baselines such as the random choice baseline or the per-category majority predictor by at least 17% in accuracy and F1. Moreover, our results corroborate past works on time series classification (Daswani et al., 2024), demonstrating that VLMs tend to outperform LLMs.

2. **Hybrid TSFM-VLM Models Are a Promising Direction For Specialized TSQA Modeling.** While hybrid TSFM-LLM approaches have previously been explored, they have only been compared against LLMs or weaker VLMs as a baseline. Our post-trained TSFM-VLM model achieves comparable overall performance in terms of accuracy and F1 with frontier models. We hypothesize that the joint training of time series encoders and VLMs is crucial for improving model performance on time series QA.

3. **Models and Domain Experts Demonstrate Complementary Abilities, Establishing a New Superhuman Frontier.** We observe substantially different error profiles between our leading

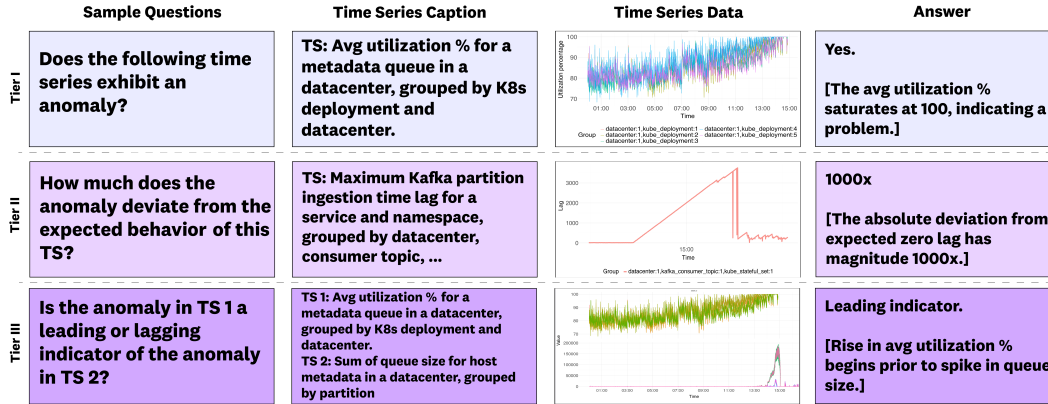


Figure 2: Example questions in ARFBench for each tier. For each question, the model is given a time series caption along with corresponding time series data in image or tabular form. In addition to the input above, time series channels are named with tags that represent collections such as datacenters, pods, services, and more.

TSFM-VLM and human experts, suggesting that their strengths are complementary. We introduce a model-expert oracle which yields 82.7% F1 and 87% accuracy, establishing a new superhuman frontier for future TSQA models.

Additionally, we analyze the impact of multimodal inputs and training stages on model performance and provide qualitative error analyses across models (Section D.5 and Appendix J). Our results suggest a promising avenue for future TSQA models, especially for observability and software incident response.

We release the benchmark data here. We also maintain a public leaderboard here.

2 ARFBENCH

2.1 PROBLEM DEFINITION

Each question in ARFBENCH consists of a templated question prompt, a caption describing one or two univariate or multivariate time series of interest, and the associated time series data. In each time series, there may be zero, one, or multiple anomalies.

Question Design. The 8 question categories are designed to test escalating levels of understanding (Figure 1). By design, as the difficulty of the question increases, the questions require more complex reasoning and additional multimodal evidence. We assign each category to a tier under this framework. The Tier I difficulty (*Presence*) consists of binary decisions that identify the presence or absence of an anomaly. The Tier II difficulty (*Identification, Magnitude, Start Time, End Time, Categorization*) corresponds to questions that require analysis of a single (possibly multivariate) time series in order to characterize specific properties of an anomaly or identify anomalous channels, thus necessitating reasoning about the time range and series values. Tier III difficulty (*Correlation, Leading/Lagging Indicator*) targets paired time series reasoning. In addition to Tier I and II-style detection and localization within each series, the model must compare anomaly structure across series—including temporal alignment and relative ordering—to determine whether anomalies are related and, if so, whether one series tends to lead or lag the other. More details about question design can be found in Appendix F.1.

Key Design Decisions. We highlight two key design components of ARFBENCH. First, we pose questions about anomalies as a classification problem, mitigating complex issues in evaluation within traditional time series anomaly detection. Second, ARFBENCH questions are designed to require strong contextual reasoning skills to relate different data modalities and prior information. More details are given in Appendix C.1.

2.2 SOFTWARE INCIDENTS AS EXPERT ANNOTATIONS.

Formally, software incidents are unplanned degradations or failures of a software system that deviate from its specified or expected behavior. Diagnosing such incidents is inherently complex, as it requires engineers to interpret noisy telemetry signals, form hypotheses, and iteratively refine their understanding of the system under evolving context.

Our primary source of data comes from software incident timelines. These timelines document the real-time process of incident investigation and resolution, including Slack Slack Technologies, LLC (2009) discussion threads between engineers at Datadog that begin when a software incident is detected and continue throughout the mitigation process. These timelines provide a real-time record of expert reasoning, including the questions engineers ask, the evidence they examine, and the conclusions they draw over time.

Interspersed with their natural language discussions, these engineers can also share embedded data visualization widgets in order to help triage and diagnose degraded systems. These visualizations encompass a wide range of telemetry data modalities, including logs, traces, and metrics. For ARFBENCH, we focus on metrics — time series data that are a key tool for anomaly reasoning and cross-metric analysis in investigations.

2.3 QA GENERATION PIPELINE

Our QA generation pipeline procedurally generates questions and uses a VLM to generate potential answer choices. We first fetch all time series contained within each incident, and run a procedural generation loop to create 8 questions, corresponding to each question category. All questions are templated and predetermined before QA generation (Table 5). Putative answer choices and correct answers are generated by prompting a VLM with the Slack discussion incident data. Finally, all QA pairs are manually filtered and corrected by the authors. More details are in Appendix C.2.

3 OVERVIEW OF RESULTS

Evaluation Metrics. We use accuracy and multiclass macro F1 to jointly measure the performance of models on ARFBench. We discuss the metrics choice in relation to ARFBench design and each metric’s strengths and weaknesses in Appendix D.1.

Baselines and Models. In Appendix D.2, we describe the types of baseline models we benchmark on ARFBENCH, consisting of simple baselines, LLMs, VLMs, and time series LLMs.

Post-Training Hybrid TSFMs. We find that time series in ARFBENCH pose various input representation problems for existing models, such as channel occlusion, lack of color variation to represent channels in time series plots, or long context length for LLMs. In order to understand how utilizing native time series modules can affect model performance, we train specialized hybrid TSFMs, where we compose existing TSFMs, VLMs, and LLMs to augment the capabilities of such existing models. We expand more on the details of the training objective, data, and architecture in Appendix D.3.

Main Takeaways. We find that (1) frontier models perform significantly better than existing baselines (Table 1). Furthermore, (2) hybrid TSFM-VLM methods are promising and show comparably strong accuracy and F1 to frontier models. Finally, we show that (3) domain experts and models demonstrate complementary strengths, establishing a new superhuman frontier for models. A more detailed analysis is given in Appendix D.4.

Ablations & Error Analysis. To understand the impact of different context components on VLMs, we conducted an ablation experiment on GPT-5, finding that additional context benefited some VLMs but hurt others (Appendix D.5). Furthermore, we find that the early stages of the hybrid TSFM-VLM hurt performance but without it, the model cannot jointly adapt to the two foundation models. Finally, we categorize reasoning errors by sampling 50 responses across all models and present several case studies (Appendix J).

Model	Accuracy (\uparrow)	Accuracy by Tier			Overall F1 (\uparrow)	F1 by Tier		
		Tier I	Tier II	Tier III		Tier I	Tier II	Tier III
<i>Baselines</i>								
Random Choice	24.5	50.0	20.0	20.0	22.5	45.6	20.5	16.8
Per-category Frequent Choice	45.1	84.7	30.1	45.6	17.3	45.9	12.3	12.5
Non-domain Experts (n=2)	69.7	80.4	63.2	72.0	60.7	68.0	59.9	59.0
Domain Experts (n=2)	72.7	89.3	67.7	71.4	64.6	76.1	64.5	60.9
Model-Expert Oracle	87.2	96.4	80.3	90.5	82.8	89.0	77.1	86.3
<i>Large Language Models (LLMs)</i>								
Qwen3 32B	47.9	80.9	35.1	48.6	36.1	55.7	31.5	33.8
GPT-5 (text)	56.4	82.6	45.2	57.9	43.8	66.1	39.6	40.3
<i>Vision-Language Models (VLMs)</i>								
Qwen3-VL 8B	45.3	80.2	40.8	37.8	34.7	63.5	36.1	23.6
Claude Sonnet 4.5	47.2	83.8	43.5	38.4	37.9	63.2	40.6	26.9
GPT-4o	47.2	79.3	49.0	34.8	42.4	64.2	43.8	33.8
GPT-4.1	47.9	80.2	50.3	34.8	44.0	65.1	48.0	33.1
Qwen3-VL 32B	52.8	80.2	46.7	49.2	45.1	65.1	41.9	41.3
Claude Opus 4.6	54.8	88.3	52.3	45.9	46.7	65.8	49.1	38.2
Gemini 3 Pro	58.1	82.9	51.0	56.5	49.6	<u>67.8</u>	49.7	43.4
GPT-5.4	61.3	81.1	<u>54.2</u>	<u>61.3</u>	<u>51.4</u>	62.6	<u>50.4</u>	48.4
GPT-5	<u>62.7</u>	82.0	55.9	<u>62.5</u>	51.9	66.9	51.2	<u>47.5</u>
<i>Post-Trained Foundation Models</i>								
OpenTSLM 1B (TS-LLM)	0.8	0.0	2.0	0.0	1.2	0.0	3.0	0.0
ChatTS 8B (TS-LLM)	31.1	60.4	26.5	25.5	22.1	48.1	20.0	15.4
Toto-Qwen3 32B (TSFM-LLM)	48.8	82.9	47.4	38.7	33.9	60.0	43.6	16.4
Qwen3-VL 32B (post-trained)	56.9	<u>84.7</u>	50.3	53.8	46.6	69.8	44.9	40.5
Toto-VLM 32B (TSFM-VLM)	63.9	<u>84.7</u>	<u>55.6</u>	64.6	48.9	66.3	48.4	43.5

Table 1: Accuracy and multiclass macro-F1 on ARFBench, sorted by model type and accuracy. Invalid answers are randomly shuffled to incorrect answer classes and F1 results are averaged over 10 random seeds. Random choice macro-F1 is averaged over 10 runs. The best model score in each metric is bolded. The second best model score in each metric is underlined.

4 LIMITATIONS AND FUTURE WORK

ARFBENCH presents a novel benchmark for evaluating the capabilities of multimodal models and introduces new modeling approaches to TSQA. However, ARFBENCH leaves out more open-ended questions commonly asked in the incident response process, such as common mitigation strategies (e.g. *Based on this anomaly, should we scale up resources?*) or linking to common incident patterns (e.g. *Is this anomaly directly linked to a deployment?*), which may require further labeled data. In addition, ARFBENCH is still limited to single-turn QA, which means it is not yet a comprehensive benchmark for agentic tasks or QA. On the modeling end, there is more work to understand why TSFM-VLM models perform so well on certain tasks and how to improve such models. We leave such tasks as future work.

5 CONCLUSION

In this work, we present ARFBENCH, a TSQA benchmark designed to test reasoning capabilities grounded in the software observability domain. We demonstrate that novel joint time series and vision-language modeling shows great promise, and we hope that the open-source release of ARFBENCH will be a valuable resource for the ML community to build upon existing methods for time series and software incident response.

REFERENCES

- Subutai Ahmad, Alexander Lavin, Scott Purdy, and Zuha Agha. Unsupervised real-time anomaly detection for streaming data. *Neurocomputing*, 262:134–147, 2017.
- Anonymous. Timeseriesexamagent: Creating timeseries reasoning benchmarks at scale. In *The Fourteenth International Conference on Learning Representations*, 2026. URL <https://openreview.net/forum?id=DewXWSvQPH>.

- Anthropic. Claude sonnet 4.5. <https://www.anthropic.com/news/claude-sonnet-4-5>, 2025. Accessed: 05/16/2025.
- Andreas Auer, Daniel Klotz, Sebastian Böck, and Sepp Hochreiter. Pre-trained forecasting models: Strong zero-shot feature extractors for time series classification. *arXiv preprint arXiv:2510.26777*, 2025.
- Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, Mei Li, Kaixin Li, Zicheng Lin, Junyang Lin, Xuejing Liu, Jiawei Liu, Chenglong Liu, Yang Liu, Dayiheng Liu, Shixuan Liu, Dunjie Lu, Ruilin Luo, Chenxu Lv, Rui Men, Lingchen Meng, Xuancheng Ren, Xingzhang Ren, Sibao Song, Yuchong Sun, Jun Tang, Jianhong Tu, Jianqiang Wan, Peng Wang, Pengfei Wang, Qiuyue Wang, Yuxuan Wang, Tianbao Xie, Yiheng Xu, Haiyang Xu, Jin Xu, Zhibo Yang, Mingkun Yang, Jianxin Yang, An Yang, Bowen Yu, Fei Zhang, Hang Zhang, Xi Zhang, Bo Zheng, Humen Zhong, Jingren Zhou, Fan Zhou, Jing Zhou, Yuanzhi Zhu, and Ke Zhu. Qwen3-vl technical report, 2025. URL <https://arxiv.org/abs/2511.21631>.
- Yifu Cai, Arjun Choudhry, Mononito Goswami, and Artur Dubrawski. Timeseriesexam: A time series understanding exam. *arXiv preprint arXiv:2410.14752*, 2024a.
- Yifu Cai, Arvind Srinivasan, Mononito Goswami, Arjun Choudhry, and Artur Dubrawski. JoLT: Jointly Learned Representations of Language and Time-Series for Clinical Time-Series Interpretation (Student Abstract). *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(21): 23447–23448, March 2024b. ISSN 2374-3468, 2159-5399. doi: 10.1609/aaai.v38i21.30423. URL <https://ojs.aaai.org/index.php/AAAI/article/view/30423>.
- Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):1–58, 2009.
- Dawei Cheng, Fangzhou Yang, Sheng Xiang, and Jin Liu. Financial time series forecasting with multi-modality graph neural network. *Pattern Recognition*, 121:108218, January 2022. ISSN 00313203. doi: 10.1016/j.patcog.2021.108218. URL <https://linkinghub.elsevier.com/retrieve/pii/S003132032100399X>.
- Winnie Chow, Lauren Gardiner, Haraldur T Hallgrímsson, Maxwell A Xu, and Shirley You Ren. Towards time series reasoning with llms. *arXiv preprint arXiv:2409.11376*, 2024.
- Ben Cohen, Emaad Khwaja, Youssef Doubli, Salahidine Lemaachi, Chris Lettieri, Charles Masson, Hugo Miccinilli, Elise Ramé, Qiqi Ren, Afshin Rostamizadeh, et al. This time is different: An observability perspective on time series foundation models. *arXiv preprint arXiv:2505.14766*, 2025.
- Jacob Cohen. A coefficient of agreement for nominal scales. *Educational and psychological measurement*, 20(1):37–46, 1960.
- Mayank Daswani, Mathias MJ Bellaiche, Marc Wilson, Desislav Ivanov, Mikhail Papkov, Eva Schnider, Jing Tang, Kay Lamerigts, Gabriela Botea, Michael A Sanchez, et al. Plots unlock time-series understanding in multimodal models. *arXiv preprint arXiv:2410.02637*, 2024.
- Felix Divo, Maurice Kraus, Anh Q Nguyen, Hao Xue, Imran Razzak, Flora D Salim, Kristian Kersting, and Devendra Singh Dhami. Quants: Question answering on time series. *arXiv preprint arXiv:2511.05124*, 2025.
- Sayna Ebrahimi, Sercan O. Arik, Yihe Dong, and Tomas Pfister. LANISTR: Multimodal Learning from Structured and Unstructured Data, April 2024. URL <http://arxiv.org/abs/2305.16556>. arXiv:2305.16556 [cs].
- Austin Feng, Andreas Varvarigos, Ioannis Panitsas, Daniela Fernandez, Jinbiao Wei, Yuwei Guo, Jialin Chen, Ali Maatouk, Leandros Tassioulas, and Rex Ying. Telecomts: A multi-modal observability dataset for time series and language analysis. *arXiv preprint arXiv:2510.06063*, 2025.

- Astha Garg, Wenyu Zhang, Jules Samaran, Ramasamy Savitha, and Chuan-Sheng Foo. An evaluation of anomaly detection and diagnosis in multivariate time series. *IEEE Transactions on Neural Networks and Learning Systems*, 33(6):2508–2517, 2021.
- 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.
- Tong Guan, Zijie Meng, Dianqi Li, Shiyu Wang, Chao-Han Huck Yang, Qingsong Wen, Zuozhu Liu, Sabato Marco Siniscalchi, Ming Jin, and Shirui Pan. Timeomni-1: Incentivizing complex reasoning with time series in large language models. *arXiv preprint arXiv:2509.24803*, 2025.
- Songqiao Han, Xiyang Hu, Hailiang Huang, Minqi Jiang, and Yue Zhao. Adbench: Anomaly detection benchmark. *Advances in neural information processing systems*, 35:32142–32159, 2022.
- Zelin He, Sarah Alnegheimish, and Matthew Reimherr. Harnessing vision-language models for time series anomaly detection. *arXiv preprint arXiv:2506.06836*, 2025.
- Alexis Huet, Jose Manuel Navarro, and Dario Rossi. Local evaluation of time series anomaly detection algorithms. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 635–645, 2022.
- Kai Kim, Howard Tsai, Rajat Sen, Abhimanyu Das, Zihao Zhou, Abhishek Tanpure, Mathew Luo, and Rose Yu. Multi-Modal Forecaster: Jointly Predicting Time Series and Textual Data, November 2024. URL <http://arxiv.org/abs/2411.06735>. arXiv:2411.06735 version: 1.
- 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.
- Herb Krasner. The cost of poor quality software in the us: A 2018 report. *Consortium for IT Software Quality, Tech. Rep.*, 10:8, 2018.
- Klaus Krippendorff. Computing krippendorff’s alpha-reliability. 2011.
- Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pp. 611–626, 2023.
- Tian Lan, Hao Duong Le, Jinbo Li, Wenjun He, Meng Wang, Chenghao Liu, and Chen Zhang. Axis: Explainable time series anomaly detection with large language models. *arXiv preprint arXiv:2509.24378*, 2025.
- Patrick Langer, Thomas Kaar, Max Rosenblattl, Maxwell A Xu, Winnie Chow, Martin Maritsch, Aradhana Verma, Brian Han, Daniel Seung Kim, Henry Chubb, et al. Opentslm: Time-series language models for reasoning over multivariate medical text-and time-series data. *arXiv preprint arXiv:2510.02410*, 2025.
- Alexander Lavin and Subutai Ahmad. Evaluating real-time anomaly detection algorithms—the numenta anomaly benchmark. In *2015 IEEE 14th international conference on machine learning and applications (ICMLA)*, pp. 38–44. IEEE, 2015.
- Shanda Li, Tanya Marwah, Junhong Shen, Weiwei Sun, Andrej Risteski, Yiming Yang, and Ameet Talwalkar. Codepde: An inference framework for llm-driven pde solver generation, 2025. URL <https://arxiv.org/abs/2505.08783>.
- Weixin Liang, Junhong Shen, Genghan Zhang, Ning Dong, Luke Zettlemoyer, and Lili Yu. Mixture-of-mamba: Enhancing multi-modal state-space models with modality-aware sparsity, 2025. URL <https://arxiv.org/abs/2501.16295>.
- Haoxin Liu, Shangqing Xu, Zhiyuan Zhao, Ling kai Kong, Harshvardhan Kamarthi, Aditya B Sasanur, Megha Sharma, Jiaming Cui, Qingsong Wen, Chao Zhang, et al. Time-mmd: A new multi-domain multimodal dataset for time series analysis. *arXiv preprint arXiv:2406.08627*, 2024.

- Haoxin Liu, Chenghao Liu, and B Aditya Prakash. A picture is worth a thousand numbers: Enabling llms reason about time series via visualization. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 7486–7518, 2025.
- Qinghua Liu and John Paparrizos. The elephant in the room: Towards a reliable time-series anomaly detection benchmark. *Advances in Neural Information Processing Systems*, 37:108231–108261, 2024.
- Ahmed Masry, Do Xuan Long, Jia Qing Tan, Shafiq Joty, and Enamul Hoque. Chartqa: A benchmark for question answering about charts with visual and logical reasoning. *arXiv preprint arXiv:2203.10244*, 2022.
- Mike A Merrill, Mingtian Tan, Vinayak Gupta, Thomas Hartvigsen, and Tim Althoff. Language models still struggle to zero-shot reason about time series. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 3512–3533, 2024.
- Jungwoo Oh, Gyubok Lee, Seongsu Bae, Joon-myung Kwon, and Edward Choi. Ecg-qa: A comprehensive question answering dataset combined with electrocardiogram. *Advances in Neural Information Processing Systems*, 36:66277–66288, 2023.
- OpenAI. Gpt-4 technical report. <https://arxiv.org/abs/2303.08774>, 2023. Accessed: 05/16/2025.
- John Paparrizos, Paul Boniol, Themis Palpanas, Ruey S Tsay, Aaron Elmore, and Michael J Franklin. Volume under the surface: a new accuracy evaluation measure for time-series anomaly detection. *Proceedings of the VLDB Endowment*, 15(11):2774–2787, 2022.
- Willa Potosnak, Cristian Challu, Mononito Goswami, Kin G Olivares, Michał Wiliński, Nina Zukowska, and Artur Dubrawski. Investigating compositional reasoning in time series foundation models. *arXiv preprint arXiv:2502.06037*, 2025.
- Penghui Qi, Zichen Liu, Xiangxin Zhou, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Defeating the training-inference mismatch via fp16. *arXiv preprint arXiv:2510.26788*, 2025.
- Sebastian Schmidl, Phillip Wenig, and Thorsten Papenbrock. Anomaly detection in time series: a comprehensive evaluation. *Proceedings of the VLDB Endowment*, 15(9):1779–1797, 2022.
- Medhasweta Sen, Zachary Gottesman, Jiaying Qiu, C Bayan Bruss, Nam Nguyen, and Tom Hartvigsen. Bedtime: A unified benchmark for automatically describing time series. *arXiv preprint arXiv:2509.05215*, 2025.
- Junhong Shen, Mikhail Khodak, and Ameet Talwalkar. Efficient architecture search for diverse tasks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.
- Junhong Shen, Tanya Marwah, and Ameet Talwalkar. UPS: Efficiently building foundation models for PDE solving via cross-modal adaptation. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL <https://openreview.net/forum?id=0r9mhjRv1E>.
- Junhong Shen, Kushal Tirumala, Michihiro Yasunaga, Ishan Misra, Luke Zettlemoyer, Lili Yu, and Chunting Zhou. Cat: Content-adaptive image tokenization, 2025. URL <https://arxiv.org/abs/2501.03120>.
- Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. In *Proceedings of the Twentieth European Conference on Computer Systems*, pp. 1279–1297, 2025.
- Haotian Si, Jianhui Li, Changhua Pei, Hang Cui, Jingwen Yang, Yongqian Sun, Shenglin Zhang, Jingjing Li, Haiming Zhang, Jing Han, et al. Timeseriesbench: An industrial-grade benchmark for time series anomaly detection models. In *2024 IEEE 35th International Symposium on Software Reliability Engineering (ISSRE)*, pp. 61–72. IEEE, 2024.
- Slack Technologies, LLC. Slack, 2009. URL <https://slack.com/>. [Computer software].

- Renbo Tu, Nicholas Roberts, Mikhail Khodak, Junhong Shen, Frederic Sala, and Ameet Talwalkar. NAS-bench-360: Benchmarking neural architecture search on diverse tasks. In *Advances in Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track*, 2022.
- Chengsen Wang, Qi Qi, Jingyu Wang, Haifeng Sun, Zirui Zhuang, Jinming Wu, Lei Zhang, and Jianxin Liao. Chatetime: A unified multimodal time series foundation model bridging numerical and textual data. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 12694–12702, 2025a.
- 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*, 2025b.
- Michał Wiliński, Mononito Goswami, Nina Żukowska, Willa Potosnak, and Artur Dubrawski. Exploring representations and interventions in time series foundation models. *arXiv preprint arXiv:2409.12915*, 2024.
- 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.
- Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Huggingface’s transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019.
- Renjie Wu and Eamonn J Keogh. Current time series anomaly detection benchmarks are flawed and are creating the illusion of progress. *IEEE transactions on knowledge and data engineering*, 35(3): 2421–2429, 2021.
- Zhe Xie, Zeyan Li, Xiao He, Longlong Xu, Xidao Wen, Tieying Zhang, Jianjun Chen, Rui Shi, and Dan Pei. Chatts: Aligning time series with llms via synthetic data for enhanced understanding and reasoning. *arXiv preprint arXiv:2412.03104*, 2024.
- Zhengzhuo Xu, Sinan Du, Yiyan Qi, Chengjin Xu, Chun Yuan, and Jian Guo. Chartbench: A benchmark for complex visual reasoning in charts. *arXiv preprint arXiv:2312.15915*, 2023.
- Zongzhe Xu, Ritvik Gupta, Wenduo Cheng, Alexander Shen, Junhong Shen, Ameet Talwalkar, and Mikhail Khodak. Specialized foundation models struggle to beat supervised baselines, 2024. URL <https://arxiv.org/abs/2411.02796>.
- Yiyuan Yang, Zichuan Liu, Lei Song, Kai Ying, Zhiguang Wang, Tom Bamford, Svitlana Vyetrenko, Jiang Bian, and Qingsong Wen. Time-ra: Towards time series reasoning for anomaly with llm feedback. *arXiv preprint arXiv:2507.15066*, 2025.
- Fangxu Yu, Hongyu Zhao, and Tianyi Zhou. Ts-reasoner: Aligning time series foundation models with llm reasoning. *arXiv preprint arXiv:2510.03519*, 2025a.
- Qiyang Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025b.
- Haochuan Zhang, Chunhua Yang, Jie Han, Liyang Qin, and Xiaoli Wang. Tempogpt: Enhancing time series reasoning via quantizing embedding. *arXiv preprint arXiv:2501.07335*, 2025a.
- Junru Zhang, Lang Feng, Xu Guo, Yuhang Wu, Yabo Dong, and Duanqing Xu. Timemaster: Training time-series multimodal llms to reason via reinforcement learning. *arXiv preprint arXiv:2506.13705*, 2025b.
- WeiQi Zhang, Jiexia Ye, Ziyue Li, Jia Li, and Fugee Tsung. DualTime: A Dual-Adapter Multimodal Language Model for Time Series Representation, July 2024. URL <http://arxiv.org/abs/2406.06620>. arXiv:2406.06620 [cs].

Yunkai Zhang, Yawen Zhang, Ming Zheng, Kezhen Chen, Chongyang Gao, Ruian Ge, Siyuan Teng, Amine Jelloul, Jinneng Rao, Xiaoyuan Guo, et al. Insight miner: A large-scale multimodal model for insight mining from time series. In *NeurIPS 2023 AI for Science Workshop*, 2023.

Zhirui Zhang, Changhua Pei, Tianyi Gao, Zhe Xie, Yibo Hao, Zhaoyang Yu, Longlong Xu, Tong Xiao, Jing Han, and Dan Pei. Timesense: Making large language models proficient in time-series analysis. *arXiv preprint arXiv:2511.06344*, 2025c.

Siru Zhong, Weilin Ruan, Ming Jin, Huan Li, Qingsong Wen, and Yuxuan Liang. Time-vlm: Exploring multimodal vision-language models for augmented time series forecasting. *arXiv preprint arXiv:2502.04395*, 2025.

Zihao Zhou and Rose Yu. Can llms understand time series anomalies? *arXiv preprint arXiv:2410.05440*, 2024.

A APPENDIX

B RELATED WORK

Evaluating Multimodal Time Series Reasoning and QA. Existing works explore multimodal time series understanding through classical tasks such as forecasting Williams et al. (2024); Liu et al. (2024); Tu et al. (2022) or in chart analysis Masry et al. (2022); Xu et al. (2023); Shen et al. (2025). Recently, TSQA benchmarks have emerged as a test of reasoning ability. Synthetic TSQA benchmarks Merrill et al. (2024); Cai et al. (2024a); Anonymous (2026); Wang et al. (2025b); Divo et al. (2025); Xie et al. (2024), where the time series is synthetic, have been a popular benchmarking approach. Real-world TSQA benchmarks are less common and often are derived from pre-existing benchmarks or datasets: ECG-QA Oh et al. (2023), TS-Insights Zhang et al. (2023), Time-mqa Kong et al. (2025), TimerBed Liu et al. (2025), and TSR-Suite Guan et al. (2025) generate QA pairs over diverse domains using publicly available time series data, but lack reasoning complexity in multivariate or multi-time series analysis and often use synthetic anomaly injection. Furthermore, many real-world TSQA datasets derive synthetic question-answer pairs without contextual evidence, i.e. there are no auxiliary human annotations that act as primary evidence for the proposed labels. Among existing benchmarks, TelecomTS Feng et al. (2025) and Time-RA Yang et al. (2025) are the most similar to this work, but neither benchmark evaluates on reasoning over multiple time series, nor are they grounded in expert annotations. Moreover, existing benchmarks often evaluate in a single-task framework, training separate models for each task. In contrast, ARFBENCH tests TSQA ability over eight different anomaly-related reasoning tasks with observability time series derived from real software incidents. Enriched with context and human annotations, the time series and QA pairs in ARFBench present a novel challenge in TSQA and reasoning, grounded in real-world and complex data.

Time Series Anomaly Detection. Time series anomaly detection is the task of proactively predicting or reactively finding significant deviations from expected behavior and trends in time series Chandola et al. (2009); Li et al. (2025); Xu et al. (2024); Goswami et al. (2022). Many benchmarks, including those grounded in observability, have been developed to assess anomaly detection Lavin & Ahmad (2015); Ahmad et al. (2017); Han et al. (2022); Schmidl et al. (2022); Si et al. (2024); Yang et al. (2025). Our work is related to, but fundamentally distinct from, classical anomaly detection benchmarks. Traditional anomaly detection formulations typically require models to output precise anomalous time ranges, and their evaluation relies on specialized metrics that attempt to reconcile ambiguity in anomaly boundaries and downstream utility (Huet et al., 2022; Garg et al., 2021; Paparrizos et al., 2022; Liu & Paparrizos, 2024). These benchmarks also commonly exhibit disagreement in ground-truth labeling, reflecting the inherent subjectivity in defining exact anomaly start and end times (Wu & Keogh, 2021). In contrast, ARFBench does not evaluate anomaly detection as a time-range prediction problem. Instead, we focus on *high-level anomaly reasoning* by framing incident analysis as a series of multiple-choice questions. Specifically, we split the task into three separate single-class classification tasks (presence, start time, end time). In doing so, we penalize performance for incorrect answers equally and thus allow a set of time ranges to equally represent the valid time range of the anomaly; the model need only find the answer that is closest in time to the ground truth. This simplified task measures reasoning relevant to downstream decisions in

Benchmark	Domain	Real-World	Context-Enriched	Multivariate	Expert-Annotated	Reasoning over Multiple Time Series
ChatTime TSQA Wang et al. (2025a)	Agnostic	✗	✗	✗	✗	✗
TimeSeriesExam Cai et al. (2024a)	Agnostic	✗	✓	✗	✗	✓
ECCG-QA Oh et al. (2023)	Medical	✓	✗	✓	✓	✓
ChatTS Xie et al. (2024)	AIOps	✗	✓	✓	✗	✓
Time-MQA Kong et al. (2025)	Multi-domain	✓	✓	✗	✗	✗
TSR-Suite Guan et al. (2025)	Multi-domain	✓	✓	✗	✗	✓
EngineMT-QA Wang et al. (2025b)	Aerospace	✗	✗	✓	✓	✓
Time-RA Yang et al. (2025)	Multi-domain	✓	✓	✓	✗	✗
Telecom-TS Feng et al. (2025)	Observability	✓	✓	✓	✗	✗
ARFBENCH	Observability	✓	✓	✓	✓	✓

Table 2: Comparison between existing time series QA benchmarks. ARFBENCH is a benchmark with real world, multivariate and context-enriched time series, and is grounded in expert annotations. Moreover, ARFBENCH also evaluates reasoning over multiple time series, which is an important task for software incident response. **Columns:** *Real-world* refers to collected or measured time series data in the application domain, as opposed to synthetic or simulated data. *Context-Enriched* refers to text input additional to (but not including) the question. *Multivariate* refers to time series with more than one channel. *Expert-Annotated* refers to human expert-authored primary source evidence analyzing the time series data. *Reasoning over Multiple Time Series* refers to question-answer pairs that include two or more different time series within the question.

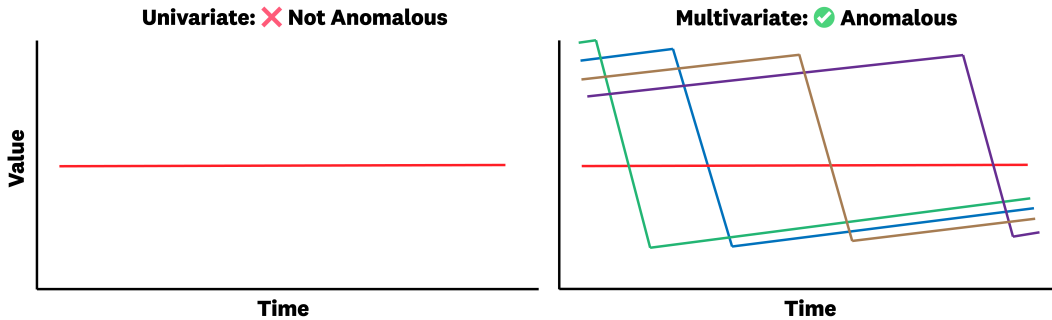


Figure 3: ARFBench requires multivariate reasoning. A single time series variate taken out of context may not appear to be anomalous. However, in a structured multivariate grouping, the variate would be considered anomalous. The time series shown above is based on average TLS certificate lifetime across cluster IDs.

incident response, where exactness of time ranges is usually not necessary. Beyond classical anomaly detection, ARFBench also includes tasks that require reasoning about anomaly types and relationships between multiple time series, such as anomaly categorization and cross-metric anomaly correlation.

Modeling Methods in TSQA. Recent works have studied how zero-shot VLMs and reinforcement learning (RL) post-trained VLMs Chow et al. (2024); Liang et al. (2025); Daswani et al. (2024); Sen et al. (2025); Liu et al. (2025); Zhang et al. (2025b); He et al. (2025) perform on time series tasks such as time series classification, TSQA, and captioning on existing time series data. VLMs have also been used for understanding anomalies in synthetic time series data Xie et al. (2024); Zhou & Yu (2024); Shen et al. (2022, 2024) as well as representation and reasoning in TSFMs Potosnak et al. (2025); Wiliński et al. (2024); Yu et al. (2025a). In parallel, multimodal time series LLM and VLM models have received increasing attention for forecasting Zhong et al. (2025); Zhang et al. (2025a); Kim et al. (2024); Cheng et al. (2022), anomaly detection and classification Zhang et al. (2024); Ebrahimi et al. (2024); Zhang et al. (2025c); Lan et al. (2025), and summarization Cai et al. (2024b); Zhang et al. (2025c). Unlike existing methods that train time series encoders from scratch, we utilize a novel approach to combine and train pretrained TSFMs with VLMs jointly, demonstrating promise in multimodal foundation model approaches to TSQA modeling. ARFBENCH is designed to allow evaluation of reasoning ability using vision, language, and time series data modalities within a well-structured TSQA framework.

C ARFBENCH DESIGN

C.1 KEY DESIGN DECISIONS

Defining Anomaly Understanding as a Classification Problem. The questions in ARFBENCH all focus on understanding time series anomalies, and therefore it is important for us to define anomalies in a consistent manner.

Historically, anomalies have been defined around scoring either points or subsequences of the time series with anomaly scores Chandola et al. (2009). Evaluation of anomaly detection has proven to be difficult, both from the benchmark and methodological side, since ambiguities exist both in terms of the ground truth (e.g. where should the start of an anomaly be?) and the evaluation metrics (e.g. how should one penalize the start of a predicted anomaly subsequence based on its distance from the start of the ground truth?). Improving evaluations in anomaly detection is still an active area of research (see Section B for more details). Here, we choose to sidestep issues with ground truth determination and choice of measure by using a multiple-choice format. In doing so, we eliminate boundary problems by constructing answer choices with a single correct choice relative to the rest.

Incorporating Additional Context is Necessary For Correct Reasoning. Anomalies are inherently contextual: whether a pattern is considered anomalous depends not only on the observed values, but also on temporal context, the semantic meaning of the signal, and its relationships to other time series. As additional context becomes available—such as longer historical windows or richer metadata about what a signal represents—the interpretation of anomalous behavior can fundamentally change. A small spike in the time series that measures unauthorized access attempts for a service may be much more severe than a brief spike in the latency of the same service. The former could suggest compromised systems, while the latter may be transient noise, with little long-term impact. Therefore, an important component of detecting and analyzing anomalies is understanding what the time series represents. In ARFBENCH, we include this crucial context by summarizing the queries used to fetch individual time series examples from a database (Figure 5, Appendix).

Moreover, multivariate groupings (Figure 3) provide insight into whether a time series variate is anomalous or not. Although multivariate anomaly detection could be done by analyzing each variate individually, this misses the broader relationship of the variate to the multivariate grouping, if structure exists. For example, a service with one pod failing and restarting may not be significant, but a combination of pods in the same service failing and restarting simultaneously may be a critical error. In ARFBENCH, all time series variates are grouped under meaningful structure, e.g., variates are grouped across specific services, datacenters, or other attributes. This requires TSQA models to reason about anomalies in a multivariate rather than univariate fashion.

C.2 QA GENERATION PIPELINE DETAILS

Our data collection process primarily involves three steps: data curation from real incidents, QA generation, and filtering and verification.

“Oracle” Model. In the construction of ARFBENCH, the discussion data is crucially used as extra context along with the time series plot as input to an “oracle” VLM to generate choices and select putative labels for questions, akin to an offline version of how expert annotations are used to label anomalies in Si et al. (2024). For particular question types, the incident timeline provides messages supporting exact answers within the text. Furthermore, having access to the incident timeline allows better understanding of the time series in comparison to other time series from the same incident. This allows for a zero-shot baseline that allows the model to “cheat”.

We use the oracle VLM for the generation, filtering, and verification steps because it allows for automated, scalable question generation when analyzing complex incident data. The incident data often requires expert domain knowledge in specific software systems or infrastructure to understand, and having the VLM synthesize such information immensely speeds up this process to generate feasible answer choices. The answers are then verified by humans by using the incident report data, which allows us to prevent leakage of sensitive data and filter the questions for erroneous answer choices.

Property	Number
Questions	750
# Distinct Timeseries	142
Data Points	5.38M
Tier I	111
Tier II	306
Tier III	333
Min # Variates	1
Median # Variates	10.5
Maximum # Variates	2283
Median Length per Variate	367
Maximum Length per Variate	40969

Table 3: Summary Statistics of ARFBENCH. ARFBench time series have high variate count and length, making input representation difficult for models.

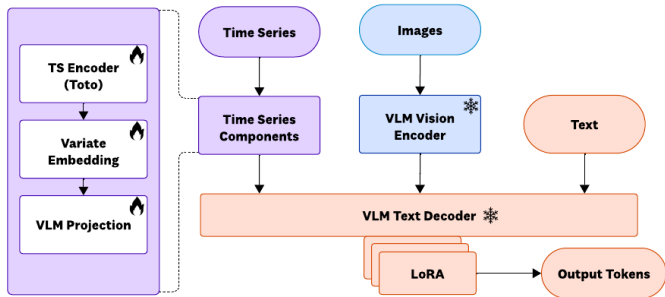


Figure 4: Architecture diagram of the Toto-VLM (Toto-Qwen3-VL) model. Frozen weights are denoted with a snowflake, while trainable weights are marked with a flame. Training a small set of weights allows us to adapt and augment the capabilities of either pretrained model.

Data Curation and Cleaning From Incidents. From each incident, we extract one or more time series to be used in QA generation. Each time series has an associated query string that maps to the given time series in a large-scale database. To remove sensitive information, we use an LLM to summarize and sanitize this query data, and we provide it as the time series caption for additional context at inference time. In addition, we include tag names corresponding to the different variates of the time series, giving context to what each channel represents (e.g. datacenter, asset, error type). See Section E.1 for further details on how the time series caption is generated.

Answer Choice and Putative Label Generation. For each time series, we generate questions using a standardized set of templates, filling in relevant time series information such as the title and channel names. Following question generation, the question, time series plot, and full incident discussion data are used as context to prompt (App. K.2) the oracle VLM to generate answer choices relevant to the particular time series, as well as to choose a putative answer choice.

Filtering and Verification. After generation, the questions are filtered by an LLM to ensure sensibility and relevance of the question-answer pairs, since some question types are more suitable for some time series than others, e.g. the question category is directly mentioned in the discussion of the incident timeline. Finally, the authors **manually filtered and corrected** all remaining QA pairs to ensure accuracy and no leakage of sensitive data.

D EXPERIMENTAL RESULTS

D.1 EVALUATION METRICS

To evaluate each model, we compute overall, per-tier, and per-category accuracy and multiclass macro-F1. Multiclass macro-F1 is the unweighted mean of the F1 score computed separately for each class, treating each class in turn as the positive label and all others as negative. Accuracy and multiclass macro-F1 serve complementary roles as evaluation metrics. Accuracy measures overall task correctness and provides an interpretable measure of benchmark-level success, while macro-F1 measures whether that performance is balanced across answer classes rather than driven by a small subset of frequent labels.

Macro-F1 prevents exploitation of accuracy. The F1 score is used in conjunction with accuracy to account for the label imbalance inherent in incident data due to the high prevalence of anomalies. Baselines such as Frequent Choice, which always answer using the most common class, can yield deceptively high accuracy on some categories (e.g., Presence) while producing uninformative predictions. Multiclass macro-F1 explicitly penalizes such predictors, achieving at most $\frac{1}{\# \text{ classes}}$.

For each question, answer choices may be data-dependent (e.g. Magnitude: 1, 10, 100, 1000), so the raw answer choices are not comparable across questions. Instead, we map raw answer choices to

predefined sets of semantic answer classes for each category in a one-to-one mapping (e.g. Smallest, Small, Medium, Large, No Anomaly). The F1 score is computed across the semantic answer choices to be comparable across classes. The semantic class definitions are given in the Appendix F.2. We find that our per-category frequent choice baseline, which chooses the most frequent label in each category (and selects randomly if the most frequent label does not exist in the answer choices), yields the second lowest F1 of all models evaluated at 17.3%.

Accuracy complements macro-F1. Macro-F1 is useful for preventing models from exploiting label imbalance, but it does not fully reflect performance on ARFBench. Since it weights all answer classes equally, macro-F1 can be sensitive to performance on rare classes and is less directly interpretable than accuracy. Furthermore, macro-F1 is non-deterministic, as we map invalid answer choices to randomly assigned incorrect classes, which can shift answers within a 0.4 pp F1 range empirically, depending on the rate of invalid answers in the responses. To mitigate this, we average the F1 results over 10 random seeds.

D.2 BASELINES AND MODEL DETAILS

Models. We use a few-shot prompting method (Appendix K.4) for all models, regardless of post-training. We refer models without post-training as “few shot”. Among few-shot models, we evaluate 9 state-of-the-art open-source and proprietary VLMs on our benchmark, as well as 2 open-source and proprietary LLMs. Furthermore, we evaluate 2 existing post-trained foundation models, both of which are time series LLM models. For each model, we generally choose the newest available checkpoint in order to test the variant with the top expected reasoning capabilities. As a simple baseline, we evaluate Random Choice as well as per-category Frequent Choice, which picks the most frequently correct answer per category if it exists in the answer choices, otherwise a randomly selected valid answer is chosen.

User Study. To compare against a human baseline, we conducted a user study with 4 time series researchers from Datadog: 2 observability domain experts and 2 non-domain experts. Users were asked to complete a randomly sampled 25% set (188 QA pairs) of the benchmark, sampled from the same question category distribution as the full benchmark. Users were given a 16-question calibration set selected from the post-training dataset, which they were required to obtain at least 90% accuracy to move on to the full user study.

Model-Expert Oracle. As a post-hoc analysis of our user study, we find that models and experts largely have different error distributions. Therefore, we posit a theoretical model-augmented expert oracle by taking the best-of-2 score between a domain expert and the Toto-VLM and having an perfect oracle judge pick the correct answer, if the correct answer is provided. (Section D.4).

D.3 TRAINING MULTIMODAL TSQA MODELS

Motivation. Time series in ARFBENCH are difficult to represent as input for few-shot LLMs and VLMs due to high variate count and length (Table 3). Furthermore, we find that existing few-shot models have major areas for improvement (Section 3). This motivates the need for specialized time series foundation models. As an initial attempt in modeling, we post-trained three model families: VLMs, TSFM-LLMs, and TSFM-VLMs. Further details on post-training can be found in Appendix H.

ARFBENCH as an RLVR Task. As defined in Guan et al. (2025), we pose ARFBENCH as a time series reinforcement learning with verifiable rewards (RLVR) task. In our setting, the model must output reasoning \hat{R} and answer \hat{y} given context (e.g. time series caption) C and up to two time series inputs $X_i = \{x_{1:T_i}^{(v_i)}\}_{v_i=1}^2$, where $i \in \{1, 2\}$ indexes the time series. V_i and T_i represent the number of channels and the length of time series i respectively. A more formal setup is given in Appendix H.3.

Reward Formulation. Given ground-truth label y , we directly use outcome reward function $R(\hat{y}, y) \in \{0, 1\}$ as an exact correctness reward based on our JSON parser. We do not give any reward for format or length, following practices in DAPO Yu et al. (2025b).

Synthetic Training Data. To post-train all models, we first generated 12,000 synthetic training examples by starting with random Gaussian noise and then injecting predefined anomalies (Figure 7).

The anomalies are injected uniformly at random into locations in the time range. Furthermore, the anomaly magnitude is scaled by a randomly chosen factor of the maximum value of the time series.

Real Training Data. We also generated and manually labeled an additional set of 207 training examples, using the same pipeline as the main benchmark, but over a date range temporally disjoint and drawn from distinct incidents than the set of incidents used for ARFBENCH. The real labeled dataset was then augmented to 395 examples with negative tier III augmentation (Appendix E.5).

TFSM-VLM Architecture. Training time series-vision-language models has only been recently explored for time series forecasting Zhong et al. (2025). Such approaches rely on training from scratch. By contrast, recent work has shown promise in using pretrained TFSMs for time series classification Auer et al. (2025). Since ARFBENCH utilizes observability metrics, we jointly train a leading observability forecasting TFSM, Toto Cohen et al. (2025), and a reasonably-sized leading open-source VLM, Qwen3-VL 32B Bai et al. (2025) (Figure 4).

To adapt Toto and Qwen3-VL to work together, we train two components to transform embeddings from Toto into the VLM embedding space. First, the variate embedding multi-layer perceptron (MLP) aggregates embeddings across the time dimension, then normalizes and projects the resulting embeddings through a simple MLP. Importantly, the variate embedding MLP prevents the textual context length from scaling with the length of the series. The VLM projection layers then project the embeddings to the hidden dimension of the text decoder.

Training Pipeline. For both the TFSM-LLM and TFSM-VLM models, we used a three-stage training approach. The models were first trained with supervised fine-tuning (SFT) on the synthetic data (Stage 1), then with SFT on the real data (Stage 2). Stage 3 continues training from the Stage 2 checkpoint using RLVR with the DAPO recipe Yu et al. (2025b), a PPO-style asymmetric clipped objective. We only use the RL stage for VLMs as it led to the best results. For each stage, we trained a different LoRA adapter for the text decoder and applied it sequentially before training the next stage. We freeze prior-stage adapters when training subsequent stages; at evaluation we compose all adapters sequentially. Although this procedure is not standard continual post-training, it enables controlled isolation of stage-specific effects and supports mixing and matching adapters across stages. For the hybrid models, we additionally train projection layers as well as finetune the TFSM backbone. Hyperparameters are listed in Table 6.

D.4 RESULTS

We present overall and per-tier accuracy and F1 scores for all models in Table 1. Per-category results can be found in Table 7 and 8 in the Appendix.

Frontier VLMs Substantially Lead Over Existing Baselines. The top few-shot model on ARFBENCH, GPT-5, yielded a 62.7% accuracy and 51.9% F1. This improves significantly over random choice, at 24.5% accuracy (22.5% F1), as well as over other open and closed baselines outside the GPT-5 family by at least 4.6 percentage points (pp) (2.3 pp F1). As expected, the models do worse as the difficulty of the question increases. Tier III the most difficult tier for all models to achieve high F1, with the best model achieving 48.4% F1. This trend tends to persist in accuracy, though, the top models tend to have higher accuracy in Tier III than Tier II. However, this can be contextualized by per-category frequent choice, showing that the label imbalance is greater in Tier III than in Tier II.

We also corroborate several previously observed trends in TSQA. First, for both LLMs we evaluated, the overall and tier-wise performance lagged behind the VLM counterpart of the same model, aligning with results from Daswani et al. (2024). Furthermore, while most VLMs are relatively strong at identifying the presence of an anomaly, they still struggle with reliably determining properties of the anomaly (Tier II) relative to humans, with a 4.6 pp (8.7 pp F1) difference between non-domain experts and GPT-5. This matches the conclusions of Zhou & Yu (2024), particularly that visual perception in vision-language models does not match human perception of anomalies.

The top proprietary and open-source few-shot VLM models also showed a substantial gap in performance, with the best open-source model (Qwen3-VL 32B) performing 9.9 pp (6.8 pp F1) worse overall compared to the best proprietary model (GPT-5.4). The main difference in performance between the two categories of models comes from Tiers II and III, where the model must reliably extract details from time series to answer the question. However, we find that some open-source models perform better than many older proprietary models or models from the Claude family.

Hybrid TSFM-VLM Models Show Promise in TSQA. Most time series LLMs feed time series as input to a time series encoder before passing the embedded input along to an LLM backbone. While these approaches may improve significantly over few-shot LLM-only inference of the base model, time series LLMs still lag significantly behind VLMs of similar size.

By contrast, our TSFM-VLM model, Toto-VLM, had comparable scores to top frontier models. Toto-VLM achieved the best accuracy (63.9%) and fourth best F1 (48.9% F1), surpassing the next best model (GPT-5) by 1.2 pp and lagging behind the best model in F1 (GPT-5) by 3 pp. Per tier, the Toto-VLM model yields the highest Tier III accuracy, as well as the second highest Tier I and Tier II accuracy. Toto-VLM also obtains comparably close tier-wise F1 scores to Gemini 3 Pro and improves significantly over the base model in all tiers, whereas other post-trained models improve in some tiers and degrade in others with respect to the base model. Analyzing the precision and recall of Toto-VLM, we find that the precision is consistently higher (2.8 pp to 15.5 pp) than the recall, suggesting that the model yields generally correct answers but under-predicts particular answer classes.

As a preliminary experiment for comparing the effect of data modalities in modeling, we compare post-trained Qwen3-VL 32B, Toto-Qwen3 32B, and Toto-VLM, which were all post-trained with the same training data, model size, and pipeline. Our results demonstrate that the time series and vision components of the architecture jointly play a large role in the performance gains of Toto-VLM, surpassing the other two models by at least 7 pp on accuracy and 2.3 pp F1 overall.

Model and Expert Performance Show Complementarity. Given the 8.8 pp accuracy and 12.7 pp F1 gap in performance between the top model and expert performance, we wanted to understand whether models and users had largely similar errors, which would imply difficult or unanswerable questions. To compare, we took the results from a top few-shot model (GPT-5 and Toto-VLM) and the two domain experts across the sampled 25% of questions used for the user study. We found that among questions that both experts got incorrect (23 questions), GPT-5 and Toto-VLM individually got 47.8% (11 questions) and 34% (8 questions) correct, respectively. Conversely, we find that among the questions that GPT-5 and Toto-VLM individually answer incorrectly (58 questions and 59 questions, respectively), at least one expert correctly answers the same questions 79.3% (46 questions) and 74.6% (44 questions) of the time. Therefore, given a large distributional mismatch between the questions models and humans answer correctly, we find evidence against unanswerable questions and an expert-level performance ceiling.

Furthermore, we find that the errors made between models and experts are qualitatively different: experts tend to make fine-grained perceptual errors and instruction-following errors that may result from the lack of interactivity of the interface and tediousness of the task, while models tend to make domain-knowledge errors and hallucinate inconsistent answers at times. We present case studies in Appendix J.2.

To establish a baseline on the performance of experts when augmented with model predictions, we compute a best-of-2 score, where we assume a perfect judge that picks a correct answer whenever presented with two plausible choices. Using the results from GPT-5 and a domain expert, we hypothesize that this is 87% accuracy and 82.7% F1, demonstrating that under ideal conditions, experts augmented with model predictions perform far better than models or humans alone.

D.5 ABLATIONS

Ablating Training Stages of Toto-VLM. As discussed in Section H, Toto-VLM is trained in three separate stages: an SFT stage with synthetic data, an SFT stage with real data, and a real data RLVR stage. We find that the performance at each stage varies heavily. The performance after the initial SFT stage is poor: a near 0 score is obtained across most questions. Since the model generates reasoning blocks without a final answer, it yields no score for a majority of questions. After the second SFT stage, Toto-VLM yields a 48.4% accuracy and 37.8% F1, which is significantly lower than the base Qwen3-VL 32B model performance (52.8% accuracy / 45.1% F1). However, this stage has relatively few answer formatting issues, suggesting that the VLM has adapted to utilizing the time series tokens. The final RLVR stage refines the model’s ability to use time series embeddings and raises the accuracy by 15.4 pp (11 pp F1) from the second SFT stage.

Model	Accuracy (\uparrow)	Accuracy by Tier			Overall F1 (\uparrow)	F1 by Tier		
		Tier I	Tier II	Tier III		Tier I	Tier II	Tier III
Random Choice	24.5	50.0	20.0	20.0	22.5	45.6	20.5	16.8
Per-category Frequent Choice	45.1	84.7	30.1	45.6	17.3	45.9	12.3	12.5
GPT-4o (No TS)	35.2	82.9	22.9	30.6	20.0	16.5	13.2	20.0
GPT-4o (No Caption)	49.3	79.3	47.7	40.8	42.0	61.0	42.3	35.4
GPT-4o (Full Input)	47.2	79.3	49.0	34.8	42.4	64.2	43.8	33.8
GPT-5 (No TS)	32.9	82.9	21.6	26.7	22.3	53.9	14.5	18.8
GPT-5 (No Caption)	56.3	80.2	49.7	54.4	46.5	64.0	46.8	40.4
GPT-5 (Full Input)	62.7	82.0	55.9	62.5	51.9	66.9	51.2	47.5

Table 4: GPT-4o and GPT-5 accuracy and F1 on ARFBench when various parts of the input are ablated. No TS data means the timeseries image was not given to the model (i.e. the answers are only dependent on the time series description). No Caption means the time series description was not given to the model (i.e. the model only has the time series plot as input). Full input means both the description and plot were present. The best performing ablation for each model is highlighted for each question category.

By contrast, we also find that the SFT stages play an important role in the joint adaptation of the TSFM and VLM. Without the SFT stages, we found that the RL training was significantly slower to improve in the first 100 steps, achieving a per-step mean reward that was around 2-4 times less than the per-step mean reward after the two SFT stages.

VLM Context Ablations. We experiment with context ablations to test the significance of each component of the multimodal input context in model performance. To do this, we test GPT-4o and GPT-5 on ARFBENCH when there are (1) no provided plots of the time series (2) no time series descriptions (captions) to condition on what the time series represents. We find that, for both models, when no time series data is provided (in image or text format), the model performs between random choice and per-category frequent choice in overall and per-tier F1 as well as accuracy. When only the time series data is provided but not the caption, GPT-5 performs substantially better (+23.4 pp accuracy / +24.2 pp F1) than GPT-5 with no time series data, but still fairly under GPT-5 with full input. By contrast, we find that GPT-4o performs better in accuracy when no caption is provided, though overall F1 still remains worse than having full input. Thus, additional context is not uniformly utilized across models in ARFBench.

E ADDITIONAL DETAILS ON BENCHMARK CONSTRUCTION PIPELINE

All questions from the benchmark are generated from 63 incidents resolved on March 7th, 2025 to March 30th, 2025. Each time series within each incident is extracted, the channel names are anonymized, and a plot is generated for the time series. Due to computational overhead, we limit the time series we extract to under 2500 non-empty variates. For each time series, all single time series question templates are applied to generate a candidate question. For each incident, we select random pairs of incidents (up to 10 pairs) and apply all paired time series question templates. For each question, we also prompt an LLM to select a putative answer. Then, all questions are filtered according to our filtering prompt, which aims to remove questions that clearly do not have overlaps in time range. The remaining questions are then uploaded to the open source labeling platform Label Studio, where the authors reviewed and changed the putative answers if necessary.

E.1 TIME SERIES CAPTION

The time series description or caption comes from summarizing database queries used to fetch the corresponding time series (Figure 5). This database query language utilizes space aggregations, time aggregations, filters, and more to select particular time series variates from a large database. In the construction of ARFBENCH, we only collect query strings from internal users and systems and do not use any customer data. These query strings are then summarized with an LLM to anonymize sensitive information and used as context in each question, as explained in Section C.2.

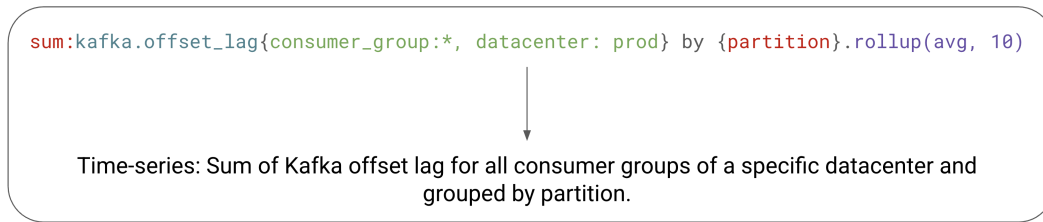


Figure 5: **Top:** Example metric query used to query the time series database. The blue text represents the metric name. The green text represents the filters. The red text represents space aggregation functions and group-by keys. Finally, the purple text represents a time aggregation function. **Bottom:** Example summarized time series description of the metric query above.

E.2 SOFTWARE INCIDENT TIMELINES IN CONTEXT OF INCIDENT RESPONSE

As mentioned, software incident timelines provide a source of expert annotations to help label our benchmark. Connecting to broader incident response workflow, the incident timeline logs the overall process of incident response, which typically consists of (but is not limited to) following nonlinear steps:

1. **Incident Report:** Alerting or triggering the beginning of the incident.
2. **Diagnosis:** Understanding the severity, systems, and customers that are affected by the incident.
3. **Triage:** Prioritizing the workflow to target different impacted components based on severity and other factors.
4. **Mitigation:** Working towards changes that reduce the negative effects of the incident.
5. **Resolution:** Finding solutions that resolve the downstream impacts of the incident.
6. **Root Cause Analysis (RCA):** Finding the specific cause of the incident.
7. **Postmortem:** Creating a write-up of what lessons were learned from the incident.

TSQA and reasoning can be a crucial tool in many of these steps, especially in triage, mitigation, and RCA. For example, looking at a time series of error rates can help to triage by identifying the portion of traffic affected. Comparing multiple time series can help identify which system component is the source of errors, aiding in RCA. Moreover, after a mitigation is attempted, responders may want to assess whether anomalous behavior in a time series has resolved to a normal state.

E.3 OBSERVABILITY TIME SERIES SUBDOMAINS

Using the captions of each time series, we classify each time series into one or more of various subdomains of observability (Figure 6). These include:

1. **Application Usage:** data collected from the application layer within a service
2. **Infrastructure:** metrics of common system measurements, such as CPU usage or GPU utilization
3. **Networking:** metrics of network- and communication-related measurements, e.g. latency
4. **Database:** data collected when performing database operations, such as querying
5. **Security:** metrics that monitor the general security of a system, such as vulnerability messages

Including each of these domains ensures a diverse benchmark with time series of diverse characteristics. To categorize the domains of the time series in ARFBENCH we use GPT-4.1 to classify each query string.

E.4 ANSWER CHOICE FILTERING

During the benchmark construction phase, the LLM pipeline will potentially generate more answer choices than are used in the final QA pair, depending on the question category. This is reflected in the

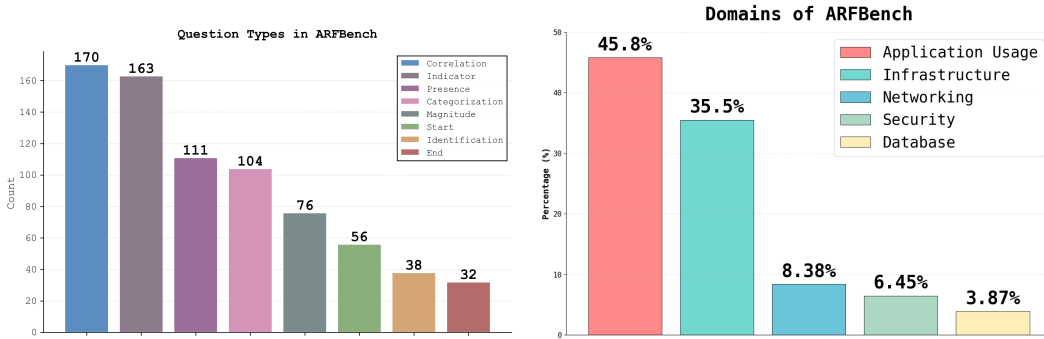


Figure 6: Distribution of question categories (left) and time series domains (right) in ARFBench.

prompts. We postprocess the benchmark to filter down to five answer choices per category, following the semantic answer classes given in Appendix F.

E.5 NEGATIVE TIER III AUGMENTATION

During question generation, questions in Tier III are generated between pairs of time series from the same incident. This often results in correlated time series, leading to a biased label distribution that may be unrepresentative of the task at hand in the real-world (i.e. the pairs of time series may not always be hand-selected by experts when a model is asked to compare them). In order to smooth the biased label distribution in Tier III after the data pipeline, we augment Tier III with questions between time series that are not in the same incident, which we determine to not be correlated or indicative of each other.

F DETAILS ON QUESTION CATEGORIES

In the following section, we provide details on the construction of each question category and associated answer choices. In all questions, an anomaly is not guaranteed to exist, and an option for “No Anomaly” is always provided. We also sort each category’s answer choices into two or five predefined semantic answer choices to keep consistent binning for multiclass F1 score.

F.1 QUESTION DEFINITIONS

Presence. The Presence question is designed to test whether a model can distinguish the existence of an anomaly in the given time series. An anomaly exists in a time series if the time series includes a subsequence of points that an expert engineer would take action upon to investigate. A time series is not limited to one anomaly, however, the question is designed to test for at least one anomaly. The answer choices for the presence question are in binary Yes/No format.

Identification. The Identification question tests whether a model can distinguish which subset of channels within the given three channels within a time series are anomalous. The given channels are chosen randomly. In total, there are five answer choices, two answer choices of single variate answer choices, one pair of variates, and the triplet of all three variates in question.

Start Time. The Start Time question tests whether a model can find the time stamp at which an anomaly begins. An anomaly may not exist within the time series. Within a given time series, the start time refers to the beginning of the first anomaly of the time series. The anomaly may begin before the start of the time series data and this case should be jointly inferred from the time series description and data. Answer choices correspond to different time stamps along the time series, with one answer as Before the earliest time stamp of the given options.

End Time. The End Time question tests whether a model can find the time stamp at which an anomaly ends. Within a given time series, the end time refers to the end of the last anomaly of the time series. The anomaly may end after the end of the time series data and this case should be jointly inferred from the time series description and data. Answer choices correspond to different time

Time Series Scope	Question Category	Difficulty	Example Template Question
Single	Presence	Tier I	Does the time series exhibit an anomaly in the given time range?
	Identification	Tier II	Which channels are exhibiting anomalies in the given time range?
	Start Time	Tier II	What is the start time of the anomaly, if an anomaly exists?
	End Time	Tier II	What is the end time of the anomaly, if an anomaly exists?
	Magnitude	Tier II	How much does the anomaly in this time-series deviate from the expected behavior of this time-series, if an anomaly exists?
	Categorization	Tier II	What type of anomaly in the given time range is exhibited, if any?
Paired	Correlation	Tier III	Does the anomaly in time-series 1 correlate with the anomaly in the other time-series, if anomalies exist?
	Leading/Lagging Indicator	Tier III	Is the anomaly in time-series 1 a leading or lagging indicator of the anomaly in time-series 2, if anomalies exist?

Table 5: An overview of the question categories within ARFBench. Questions are ordered by increasing difficulty, corresponding to the required reasoning and synthesis across all given inputs. Each increasing tier of difficulty requires the correct reasoning of the previous lower tier.

stamps along the time series, which are not guaranteed to lie within the time range of the time series, with one answer as Not resolved referring to the end time of the anomaly being after the end of the given time series data.

Magnitude. The Magnitude question tests whether a model can correctly assess the deviation of the time series from an expected counterfactual forecast. In general, this is calculated as the maximum ratio between the anomalous values and the mean counterfactual values. In the case that the counterfactual values are zero, the magnitude is instead calculated as the maximum absolute value of the anomalous values. For this question, the LLM is prompted to generate reasonable answer choices in a logarithmic scale.

Categorization. The Categorization question tests whether a model can correctly classify an anomaly into one of five anomaly categories relevant for software incident response, which include: level shift, transient spike, change in seasonality, change in variance, and change in trend. We qualitatively describe these categories in relation to the traditional categories of anomalies in anomaly detection Chandola et al. (2009) (i.e. point, contextual, and collective anomalies), but we find that the granularity of the definitions for anomaly detection is not sufficient for analyzing anomalies relevant in software incident response.

A level shift represents a prolonged change in mean value of the time series. The level shift may arise as a contextual anomaly, where the mean value is substantially different than what is expected from the general value of the time series in relation to its time series description, or it may be a collective anomaly, in which we see an identifiable subsequence of the data have a change in mean value over time.

A transient spike is a type of point anomaly, where a small collection of points are substantially different in magnitude than other points in the time series. Transient spikes often return to the original values and are very short in duration, leading to little to no shift in mean value.

A change in seasonality is a type of contextual or collective anomaly, where the magnitude of the time series does not change, but the seasonal pattern of the time series changes in frequency.

A change in variance describes when the variance of the time series significantly shifts from its previous value for a sustained period of time.

A change in trend is a contextual or collective anomaly in which the time series increases or decreases for a significant period of time.

Multiple such anomalies may occur in one series, but in general, we consider the one with the largest magnitude or most relevant to the incident timeline to be the correct answer.

Correlation. The Correlation question tests whether a model correctly identifies correlations between two time series as well as a model can correctly identify the existence of anomalies in both. Two anomalies are correlated if they have temporally similar patterns in context to the time series descriptions, or if they have known relation through the software infrastructure based on what they represent.

Indicator. The Indicator question is a follow-up question the correlation question, testing a model on whether it can identify the temporal order between anomalies. The answer choices include that the first anomaly is a leading indicator of the second anomaly, a lagging indicator of the second anomaly, perfectly correlated between the two anomalies, or not correlated at all.

F.2 SEMANTIC BINNING CATEGORIES

Presence. The semantic classes of Presence are the same as the answer choice classes (i.e. Yes/No).

Identification. The semantic classes of Identification are No Anomaly (in the channels listed), One Channel (small), One Channel (large), Two Channels, Three Channels, with the single variate answer choices ordered lexicographically.

Start Time. The semantic classes of Start Time are No Anomaly, Earliest, Early, Medium, Late, corresponding to the temporal order of the answer choices presented.

End Time. The semantic classes of End Time are No Anomaly, Early, Medium, Late, Latest, corresponding to the temporal order of the answer choices presented.

Magnitude. The semantic classes of End Time are No Anomaly, Smallest, Small, Medium, Large, corresponding to the magnitude order of the answer choices presented.

Categorization. The semantic classes for Categorization are No Anomaly, Change in Trend, Transient Spike, Level Shift, Change in Seasonality/Variance.

Correlation. The semantic classes match the answer choices exactly. These include: No, there is no anomaly in either time-series, No, there is an anomaly in both but they are not correlated, No, there is an anomaly only in time-series 1, No, there is an anomaly only in time-series 2, Yes, there is an anomaly in both and they are correlated.

Indicator. The semantic classes match the answer choices exactly. These include: The anomaly in time-series 1 is a leading indicator of the anomaly in time-series 2, The anomaly in time-series 1 is a lagging indicator of the anomaly in time-series 2, The anomaly in time-series 1 is not correlated to the anomaly in time-series 2, The anomaly in time-series 1 is perfectly correlated to the anomaly in time-series 2, No Anomaly in one or both series.

G ADDITIONAL DETAILS ON LLM SETTINGS FOR INFERENCE

Inference. To evaluate each of the foundation models, we used one of two methods. First, closed-source models were accessed through their respective APIs: Claude Sonnet 4.5 and Claude Opus 4.6 was evaluated through the Anthropic API Anthropic (2025), GPT-5, GPT-5.4, GPT-4.1, and GPT-4o

were evaluated with the OpenAI API OpenAI (2023), and Gemini 3 Pro Preview was evaluated through the Google Gemini API.

Open-weight models including Qwen3-VL 8B, 32B, Qwen3 32B were evaluated via the vLLM library Kwon et al. (2023), while models such as ChatTS, OpenTSLM, Toto-Qwen3, Toto-VLM were evaluated with Transformers. All models were evaluated with eight NVIDIA A100 GPUs with tensor parallel, using weights downloaded from Hugging Face Transformers Wolf et al. (2019).

In terms of model choice, we preferred newer models due to higher overall accuracy on multimodal benchmarks. For the OpenTSLM series, we choose the soft-prompt (SP) Llama-3.2 1B model due to its high test performance on benchmarks reported in the OpenTSLM paper Langer et al. (2025).

For inference, we use temperature 0.05 for all third-party models for factuality and determinism (except GPT-5, GPT-5.4, Gemini 3 Pro, which had default temperature 1.0) with maximum token output of 1000 tokens. For reasoning models, we set their reasoning level to “medium” with 2000 tokens or “high” with 10000 tokens and we take the best performing model. In cases where reasoning models gave blank answers due to reaching or surpassing the maximum output tokens, we labeled such answers as incorrect. For the Toto-VLM and Toto-Qwen3 models, we use temperature 0.

Due to context length and channel quantity limits on ChatTS, we make several preprocessing decisions before inputting the time series to the model. To process time series longer than 1024, we subsample the time series down to 1024. For time series with more than 50 variates, we select the 50 variates with the highest mean from each time series in the case of single time series questions, and the 25 variates with the highest mean from each time series in the case of paired time series questions. For the GPT-5 text and Qwen3 32B models, which have a longer context length, we first find a single optimal interval for which the entire series can fit in the context length, and if not, we instead truncate a fraction of the time series data from the middle of the series at various truncation ratios (25%, 40%, 55%, 70%) until we achieve the desired context.

Multiple Choice Shuffling. At inference time, we randomly shuffle the answer choices before building the prompt in order to prevent option order bias that may result from any one instantiation of ARFBench.

Plot Generation. In order to evaluate vision-language models with time series data, we transform each time series into a plot with Matplotlib (for single series) and Plotnine (for paired series) with 100 dots per inch (DPI) and then downsize each plot to have a maximum size of 1500 pixels on each side in order to satisfy API requirements. For Qwen3-VL we allow models to auto-resize the images using the built-in Qwen utility tools. For Tier I and Tier II images, the model receives one image as input, corresponding to the time series. We only include channel names in the plot for Identification questions. In Tier III, we input three images: the first image has both time series stacked vertically with a shared time axis. The channels are monochromatic, i.e. all channels of the first time series are blue and all channels of the second time series are orange. The last two plots are plots of each individual time series, in the order they are referenced in the question.

Furthermore, the channel names of the time series are omitted from the time series plot in order to retain image resolution and clarity, unless there are less than 8 channels. The channel names are provided in other components (e.g. prompt or question) of the textual input for all QA pairs.

H ADDITIONAL DETAILS ON TRAINING

We trained three types of models: vision-language models, TSFM-LLM models, and TSFM-VLM models.

H.1 DATA

Both synthetic data and real data were used for post-training. Synthetic data was generated by generating scaled random Gaussian noise time series of various lengths, from 240 to 50,000 timestamps, with 40% of series distributed within 1000-10,000 timestamps. The time series are randomly perturbed with seasonal and drift components for diversity. We predefine several anomaly types, including level shifts, transient spikes, changes in variance, seasonality, and flatline anomalies. For question generation, we generate an equal number of each category, randomly selecting an anomaly (or no



Figure 7: Process of generating synthetic multivariate data for post-training VLMs, TSFM-VLMs, and TSFM-LLMs. Gaussian noise is transformed by adding seasonality and drift components, then anomalies are injected at randomly chosen points within the time series. To make the data more realistic, captions, channel names, and reasoning are generated with an LLM to reflect the observability setting.

anomaly at all) to inject into the synthetic time series. The timestamp of anomaly injection is also uniformly chosen across the length of the time series. We generated a total of 12,000 synthetic QA pairs for our training set.

The real data was generated with the ARFBench pipeline on incidents resolved on April 1st to April 7th, 2025. The incidents were manually cleaned to ensure no data overlap with the benchmark data. A total of 207 examples were labeled, which was then augmented to 395 examples via Tier III augmentation.

For both sets of training data, synthetic reasoning traces were added by prompting a VLM for a reasoning explanation given the question and correct answer. These traces were used as supervision for both SFT stage 1 and stage 2.

H.2 TRAINING FRAMEWORKS AND HARDWARE

VLM post-training was done with the veRL Sheng et al. (2025) framework with 8 NVIDIA H200s, with training and inference done in the FP16 precision Qi et al. (2025). Toto-Qwen3 and Toto-VLM models were trained with TRL and patched to support time series input. Training was done with 8 NVIDIA B200s for Toto-Qwen3 and Toto-VLM. For Toto related models, some additional data was provided in the prompt for context: the start and end absolute timestamp, the timestep interval, and variate names in the order they were provided to Toto.

H.3 RL FORMULATION.

Recall our time series RLVR setting: for context (including prompt, question, and additional context) C and time series input $\{X_j\}_{j=1}^k$ for $k \in \{1, 2\}$, we sample a group G of outputs $\{o_i\}_{i=1}^n \sim \pi_{\theta_{\text{old}}}(\cdot |$

Hyperparameter	VLM	TSFM-LLM	TSFM-VLM
<i>Common</i>			
Optimizer	AdamW	AdamW	AdamW
Precision (training & inference)	fp16	bf16	bf16
Number of devices	8	8	8
<i>LoRA / adapters</i>			
Trainable parameters	LoRA only	LoRA + Toto + proj	LoRA + Toto + proj
Freeze vision tower	True	True	True
Freeze TS Encoder	–	False	False
<i>SFT Stage 1 (synthetic)</i>			
Used?	No	Yes	Yes
Epochs	–	3	3
Learning rate	–	1e-05	1e-05
Batch size (global)	–	8	8
Warmup steps	–	10	10
Weight decay	–	1e-2	1e-2
Max time series length	–	51200	51200
Max variates	–	100	100
<i>SFT Stage 2 (real)</i>			
Used?	No	Yes	Yes
Epochs	–	4	4
Learning rate	–	1e-05	1e-05
Global Batch size	–	8	8
Warmup steps	–	10	10
Weight decay	–	1e-2	1e-2
Max time series length	–	25600	25600
Max variates	–	100	100
<i>RL Stage 3 (DAPO)</i>			
Max context length	20000	20000	20000
Global batch size	16	40	32
Generations per prompt (n)	8	8	8
ϵ_{low} (DAPO low clip coefficient)	0.2	0.2	0.2
ϵ_{hi} (DAPO high clip coefficient)	0.28	0.28	0.28
Policy learning rate	1e-06	2e-05	1e-05
Max time series length	–	25600	25600
Max variates	–	100	100
Temperature	–	1.3	0.8

Table 6: Post-training hyperparameters for SFT and RL (DAPO) across model families.

$C, \{X_j\}_{j=1}^k$), where the output $o_i = (\hat{R}_i, \hat{y}_i)$ includes both reasoning and model answer. For each group, each output o_i receives a reward r_i as defined by our 0-1 reward function in Section ??.

DAPO Yu et al. (2025b) optimizes our multimodal models π_θ with the following objective:

$$\mathcal{J}(\theta) = \frac{1}{n} \sum_{i=1}^n \frac{1}{|o_i|} \sum_{q=1}^{|o_i|} \min\left(\phi_{i,q}(\theta) \hat{A}_i, \text{clip}(\phi_{i,q}(\theta), 1 - \epsilon_{low}, 1 + \epsilon_{hi}) \hat{A}_i\right),$$

where $\phi_{i,q}(\theta)$ is the token-level importance ratio

$$\phi_{i,q}(\theta) = \frac{\pi_\theta(o_{i,q} | C, \{X_j\}_{j=1}^k, o_{i,<q})}{\pi_{\theta_{old}}(o_{i,q} | C, \{X_j\}_{j=1}^k, o_{i,<q})},$$

and \hat{A}_i is the group-relative advantage computed from rewards:

$$\hat{A}_i = \frac{r_i - \text{mean}(\{r_i\}_{i=1}^n)}{\text{std}(\{r_i\}_{i=1}^n)}.$$

Model	Overall Accuracy (↑)	Accuracy by Category							
		Tier I		Tier II				Tier III	
		Presence	Identification	Start Time	End Time	Magnitude	Categorization	Correlation	Indicator
<i>Baselines</i>									
Random Choice	24.5	50.0	20.0	20.0	20.0	20.0	20.0	20.0	20.0
Per-category Frequent Choice	45.1	84.7	36.8	35.7	34.4	17.1	32.7	42.9	48.5
Non-domain Experts (n=2)	69.7	80.4	66.7	64.3	68.8	60.5	61.5	72.1	72.0
Domain Experts (n=2)	72.7	89.3	77.8	67.9	75	60.5	72.4	74.4	68.3
Model-Expert Oracle	87.2	96.4	77.8	78.6	100.0	68.4	84.6	95.4	85.4
<i>Large Language Models (LLMs)</i>									
Qwen3 32B (text)	47.9	80.9	28.9	27.3	35.5	37.3	39.8	50.9	46.3
GPT-5 (text)	56.4	82.6	<u>47.4</u>	29.6	38.7	51.4	50.0	56.9	59.0
<i>Vision-Language Models (VLMs)</i>									
Qwen3-VL 8B	45.3	80.2	26.3	25.0	31.3	57.9	45.2	57.1	17.8
Claude Sonnet 4.5	47.2	83.8	18.4	30.4	37.5	53.9	53.8	58.8	17.2
GPT-4o	47.2	79.3	39.5	35.7	43.8	61.8	51.9	45.3	23.9
GPT-4.1	47.9	80.2	28.9	33.9	40.6	68.4	56.7	45.9	23.3
Qwen3-VL 32B	52.8	80.2	23.7	33.9	56.3	59.2	50.0	61.8	36.2
Claude Opus 4.6	54.8	88.3	31.6	37.5	53.1	57.9	63.5	65.9	25.2
Gemini 3 Pro	58.1	82.9	28.9	44.6	62.5	56.7	54.8	71.2	41.1
GPT-5.4	61.3	81.1	31.6	63.6	<u>65.6</u>	57.9	56.7	61.8	<u>60.7</u>
GPT-5	<u>62.7</u>	82.0	31.6	44.6	68.8	<u>65.8</u>	59.6	63.5	61.3
<i>Post-Trained Foundation Models</i>									
OpenTSLM (TS-LLM)	0.8	0.0	0.0	3.6	0.0	5.3	0.0	0.0	0.0
ChatTS (TS-LLM)	31.1	59.5	15.8	16.1	15.6	28.9	20.2	40.0	14.7
Toto-Qwen3 (TSFM-LLM)	48.8	82.9	10.5	35.7	34.4	47.4	71.2	41.8	35.6
Qwen3-VL 32B (post-trained)	56.9	<u>84.7</u>	36.8	41.1	43.8	63.2	52.9	<u>67.6</u>	39.3
Toto-VLM (TSFM-VLM)	63.9	<u>84.7</u>	47.4	26.8	59.4	<u>64.5</u>	<u>66.3</u>	<u>68.8</u>	<u>60.1</u>

Table 7: Accuracy on ARFBench across question categories and models. For Random Choice only, invalid answers are randomly shuffled to incorrect answer classes. The best model performance in each metric is bolded. The second best model performance in each metric is underlined. Our model, Toto-VLM, achieves the top accuracy among models tested.

Model	Overall F1 (↑)	F1 by Category							
		Tier I		Tier II				Tier III	
		Presence	Identification	Start Time	End Time	Magnitude	Categorization	Correlation	Indicator
<i>Baselines</i>									
Random Choice	22.5	45.6	21.2	18.9	18.2	20.4	21.7	15.8	17.8
Per-category Frequent Choice	17.3	45.9	10.8	16.3	14.1	6.0	14.6	12.0	13.1
Non-domain Experts (n=2)	61.3	68.0	79.0	67.4	67.2	40.3	61.2	58.4	62.4
Domain Experts (n=2)	64.6	76.1	77.5	74.2	72.6	51.8	67.3	64.1	57.6
Model-Expert Oracle	82.8	89.0	68.3	83.4	1.0	67.0	75.6	94.4	77.8
<i>Large Language Models (LLMs)</i>									
Qwen3 32B	36.1	55.7	28.4	26.6	26.9	31.4	36.8	32.3	35.4
GPT-5 (text)	43.8	66.1	38.1	27.9	27.0	44.8	47.6	38.0	42.4
<i>Vision-Language Models (VLMs)</i>									
Qwen3-VL 8B	34.7	63.5	28.6	21.8	23.5	47.0	42.8	33.1	13.8
Claude Sonnet 4.5	37.9	63.2	16.8	33.2	31.3	49.3	49.8	33.8	19.8
GPT-4o	42.4	64.2	34.6	30.3	36.1	51.8	50.8	40.1	27.2
GPT-4.1	44.0	65.1	29.2	33.5	32.7	63.7	55.9	42.9	23.3
Qwen3-VL 32B	45.1	65.1	25.0	30.8	46.7	46.9	49.0	47.5	34.7
Claude Opus 4.6	46.7	65.8	34.3	36.1	45.1	53.8	<u>59.2</u>	51.6	24.1
Gemini 3 Pro	49.6	<u>67.8</u>	38.6	43.3	57.1	50.3	54.5	57.0	29.2
GPT-5.4	<u>51.4</u>	62.6	29.6	53.3	<u>55.1</u>	51.7	54.1	47.7	49.1
GPT-5	51.9	66.8	32.8	<u>44.2</u>	47.8	<u>59.1</u>	57.0	49.0	<u>45.9</u>
<i>Post-Trained Foundation Models</i>									
OpenTSLM 1B (TS-LLM)	1.2	0.0	8.2	2.7	0.0	6.0	0.0	0.0	0.0
ChatTS 8B (TS-LLM)	22.1	48.1	22.2	15.0	14.4	27.9	17.9	21.4	9.2
Toto-Qwen3 32B (TSFM-LLM)	33.9	59.9	17.5	41.3	23.0	35.9	66.2	18.6	14.1
Qwen3-VL 32B (post-trained)	46.6	69.7	<u>40.5</u>	37.2	36.7	48.9	50.3	46.8	33.9
Toto-VLM 32B (TSFM-VLM)	48.9	66.3	46.9	23.0	48.8	54.1	58.4	44.2	42.7

Table 8: Multiclass F1 on ARFBench across question categories and models. Invalid answers are randomly shuffled to incorrect answer classes and results are averaged over 10 random seeds where applicable. The best model performance in each metric is bolded. The second best model performance in each metric is underlined.

I ADDITIONAL RESULTS

I.1 PER CATEGORY RESULTS

The per-category accuracy and F1 results are shown in Tables 7 and 8, respectively.

J ERROR ANALYSIS AND CASE STUDIES

We qualitatively analyze the errors made by the models on the benchmark to further understand model output. To do this, we have the model output a reasoning explanation. We sample 50 such

Model	Cohen’s Kappa (\uparrow)	Krippendorff’s Alpha (\uparrow)
Domain Expert 1 and Expert 2	0.667	0.672
Non-domain Expert 1 and Non-domain Expert 2	0.627	0.626
Domain Expert 1 and Non-domain Expert 1	0.674	0.673
Domain Expert 1 and Non-domain Expert 2	0.666	0.667
Domain Expert 2 and Non-domain Expert 1	0.700	0.705
Domain Expert 2 and Non-domain Expert 2	0.711	0.715

Table 9: Cohen’s Kappa and Krippendorff’s alpha values for inter-annotator across domain experts and non-domain experts. We find that inter-annotator agreement is generally substantial for all groups for Cohen’s kappa, and tentatively acceptable for most pairs for Krippendorff’s alpha.

errors across all models and categorize them into three different error categories: incorrect perception, instruction following errors, and limited context usage errors.

Incorrect Perception (48%). These errors result from incorrect analysis of the time series. This includes hallucinating time series values or time ranges, identifying the wrong anomaly time range, or reasoning about incorrect details regarding the time series itself.

Limited Context Usage (42%). These errors result when not all provided information is utilized. For example, models may neglect processing all channels, leading to information loss. Without utilizing all channels, the model may not be able to contextualize the anomaly properly. In general, this can include incorrect multivariate understanding, lack of caption usage, or lack of domain knowledge.

Instruction Following Errors (10%). These errors occur from incorrect instruction following. For example, in many Correlation questions, models answer “The anomaly in time series 1 is not correlated to the anomaly in time series 2” even after deducing that there is no anomaly one of the time series.

We highlight several case studies (Figures 8, 9, 10, 11) and present them below.

J.1 INTER-ANNOTATOR AGREEMENT BETWEEN USERS

We calculate the inter-annotator agreement using Cohen’s Kappa Cohen (1960) and Krippendorff’s alpha Krippendorff (2011), two commonly used robust statistics for inter-annotator agreement, across all pairs of annotators (Table 9). We find that inter-annotator agreement is generally substantial for Cohen’s Kappa across pairs ($0.6 \leq \kappa \leq 0.8$), and tentatively acceptable ($\alpha \geq 0.667$) for Krippendorff’s alpha for all pairs except between the non-domain experts.

J.2 QUALITATIVE COMPARISON OF MODEL ERRORS WITH DOMAIN EXPERTS

Analyzing the specific questions that models and experts incorrectly answer, we find that experts tend to make errors on fine-grained perception and some instruction following errors, while models tend to make errors on hallucinations and utilizing domain knowledge.

We present five example questions that describe some of the differences in errors that are made between models and domain experts in Figures 12, 13, 14, 15.

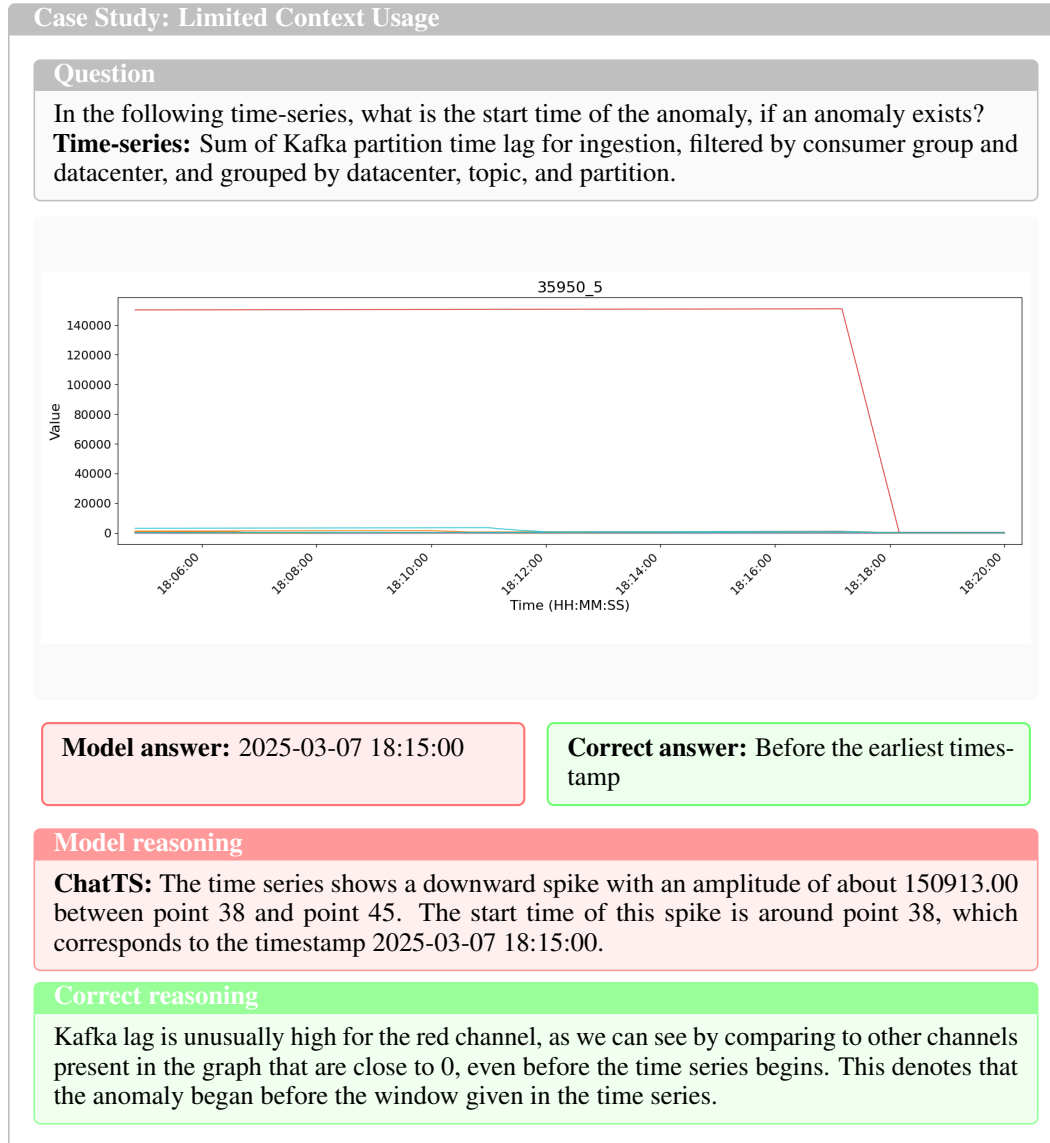
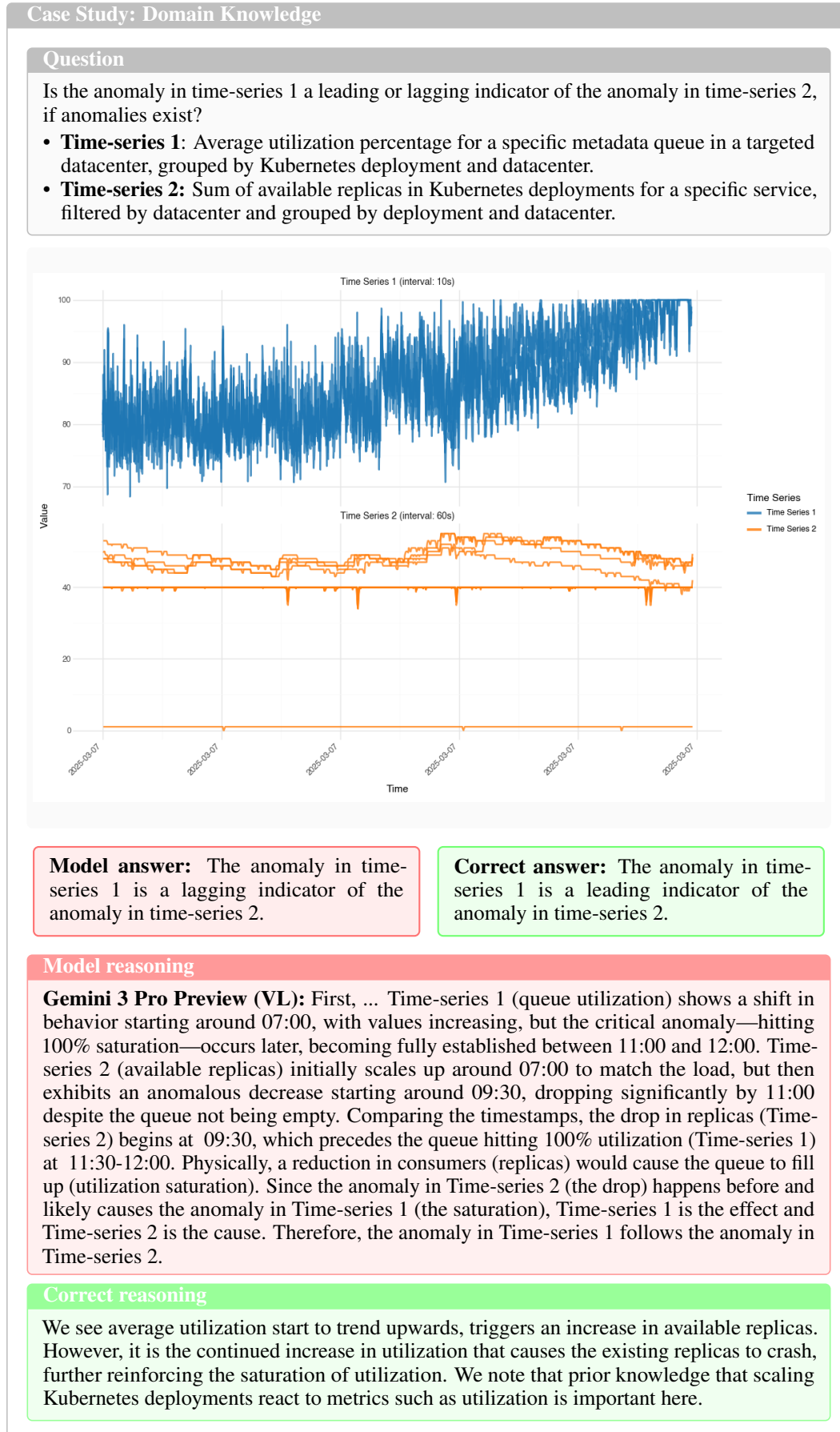


Figure 8: An example where ChatTS does not utilize the extra textual context given in the question to solve the problem.



Figure 9: An example where GPT-4.1 (VLM) lacks the domain knowledge and perceptual ability to notice an anomaly.



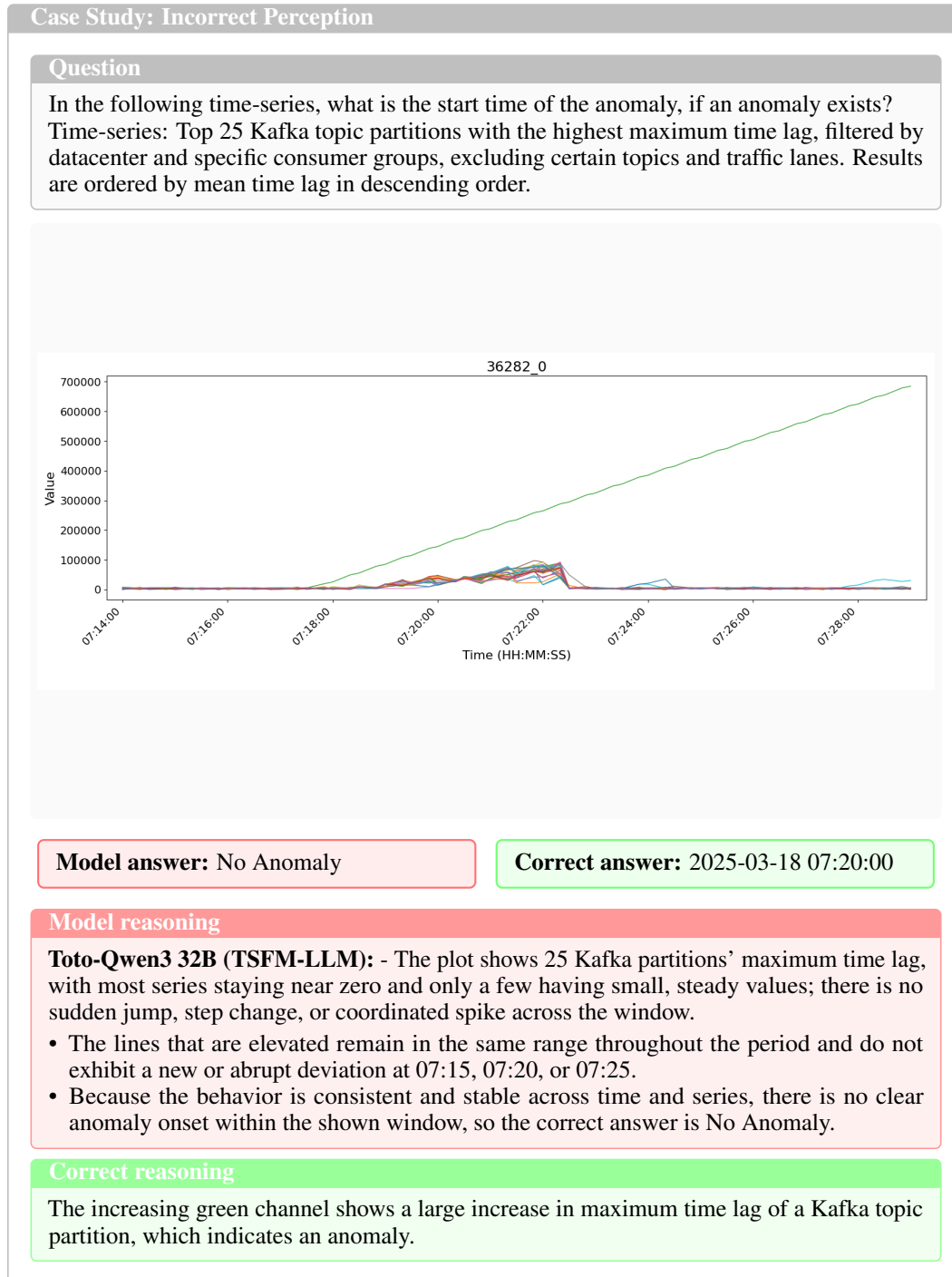


Figure 11: Toto-Qwen3 32B does not correctly perceive the time series, missing the changing trend in the green channel.

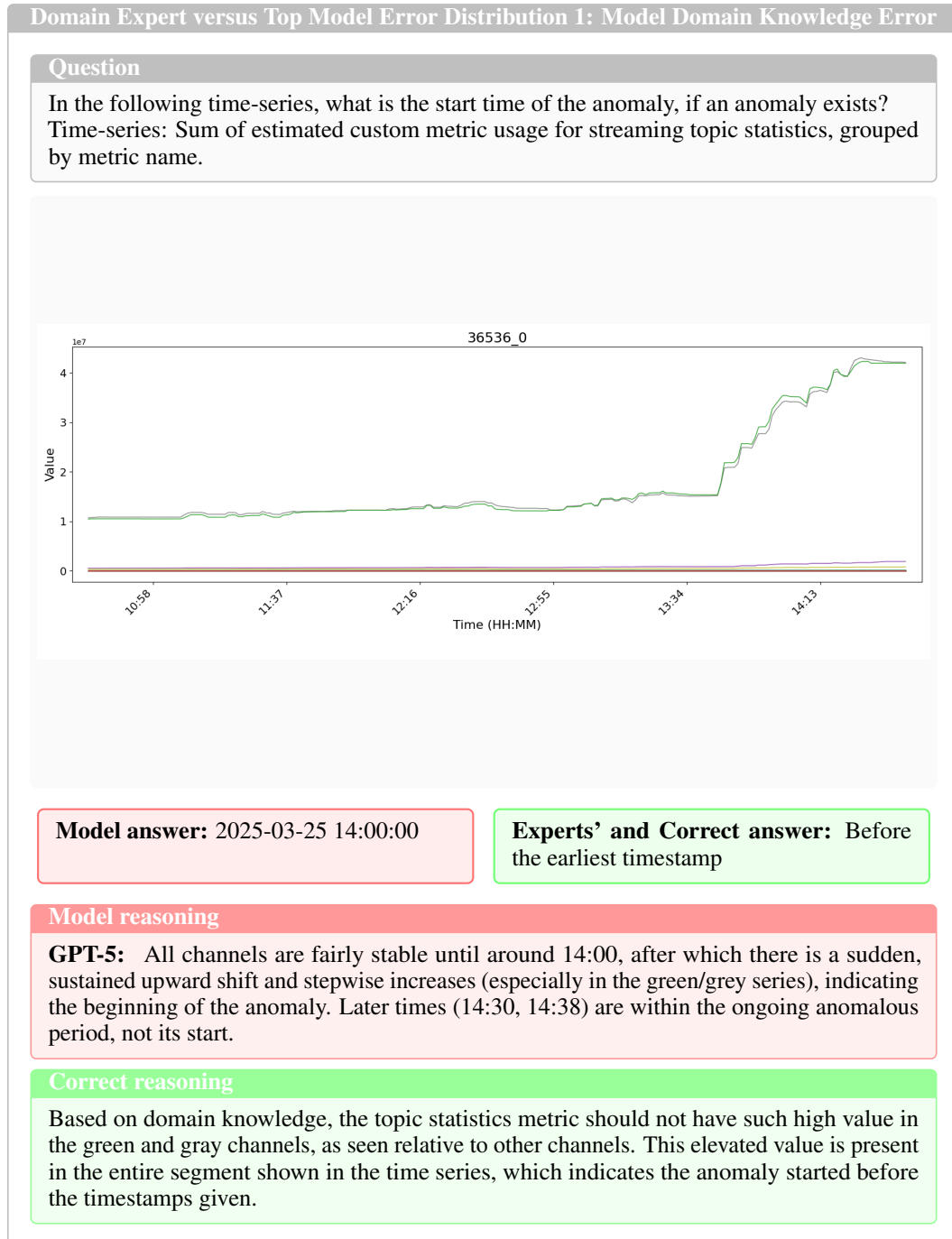


Figure 12: Top models fail to use sufficient domain knowledge, in contrast to domain experts.

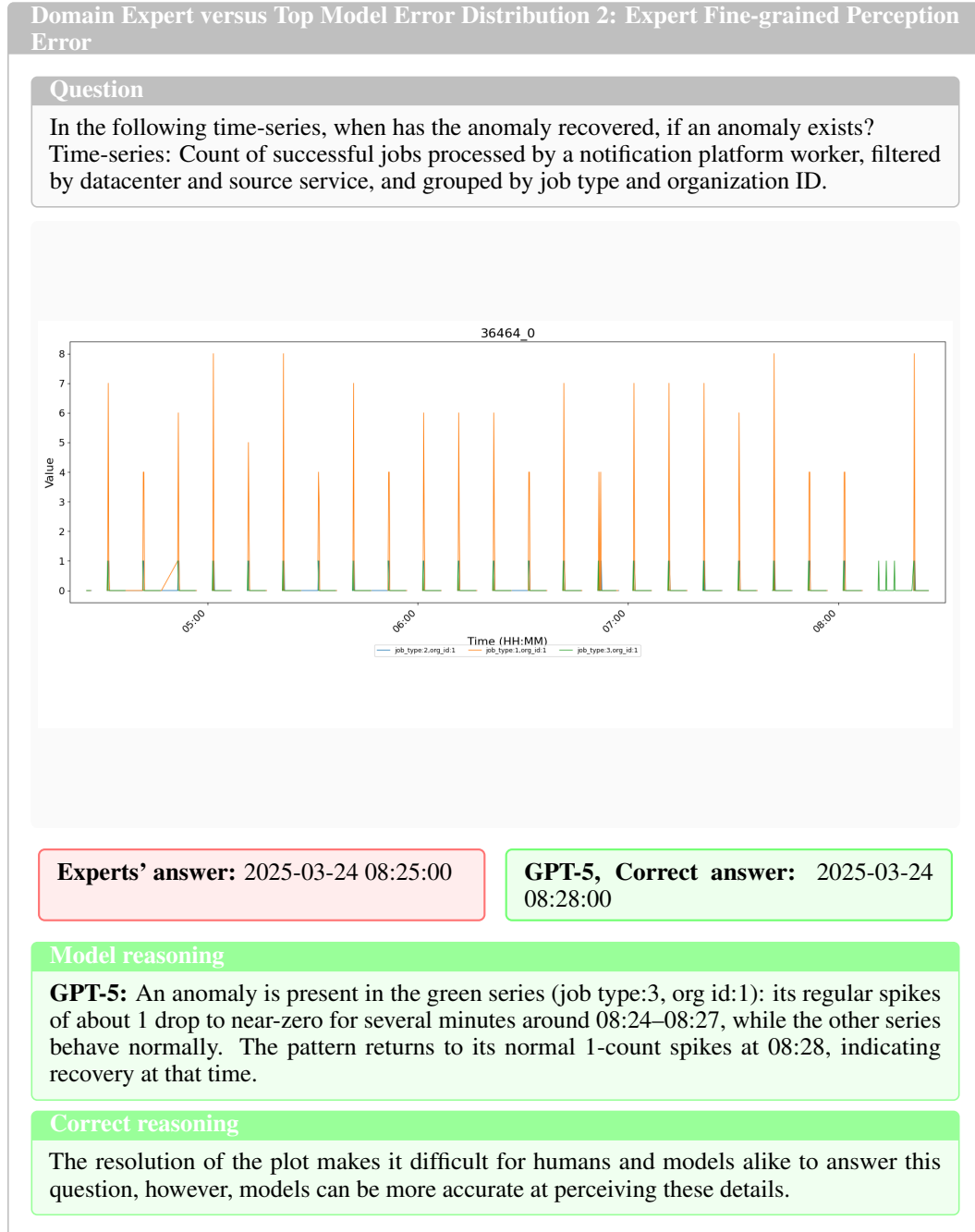


Figure 13: Experts do not always perceive fine-grained details about exact timings of when anomalies start or end, leading to perceptual errors, in contrast to models.

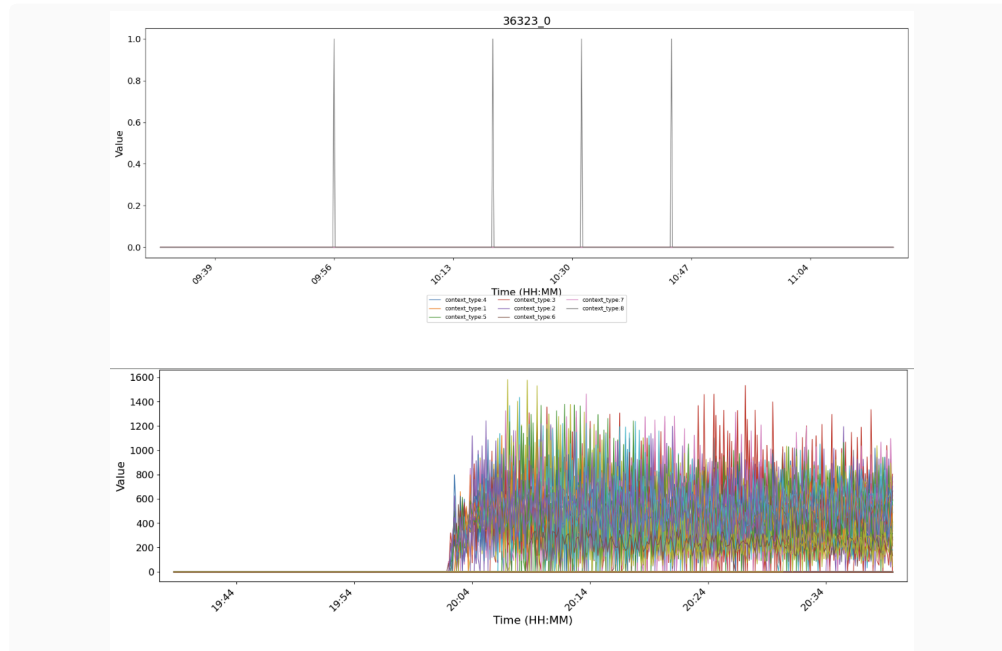
Domain Expert versus Top Model Error Distribution 3: Expert Understanding/Instruction-Following Error

Question

In the following time-series, does the anomaly in this time-series correlate with the anomaly in the other time-series, if anomalies exist?

Time-series 1: Count of build failures for a specific event context provider service in a production datacenter, grouped by context type.

Time-series 2: Rate of indexer errors in an orchestration system for a specific release group, filtered by various dimensions and grouped by pod.



Expert Answer 1: No, there is an anomaly only in time-series 2

Expert Answer 2: Yes, there is an anomaly in both and they are correlated

GPT-5, Toto-VLM, and Correct answer: No, there is an anomaly in both but they are not correlated

Model reasoning

Toto-VLM 32B (TSFM-VLM): Time-series 1 shows a single isolated spike on 2025-03-20, while time-series 2 has a large spike on 2025-03-25; these occur on different dates and do not overlap. The third plot has a clear surge beginning around 19:59 that grows into a sustained high-variance burst from 20:04 to 20:35, which is a classic ramp-up followed by a plateau. Because the spikes are separated by days and have different shapes (isolated vs ramping), there is no temporal alignment or co-movement between the series, so both have anomalies but they are not correlated.

Correct reasoning

The time series displayed are on different days and are temporally unrelated (there is another plot showing no temporal overlap that is elided due to space constraints). However, they both display anomalies relevant to their respective metrics (failures and errors). The TSFM-VLM hallucinates some details but gets the reasoning correct.

Figure 14: Experts also make some understanding and/or instruction-following errors which models often do not make.

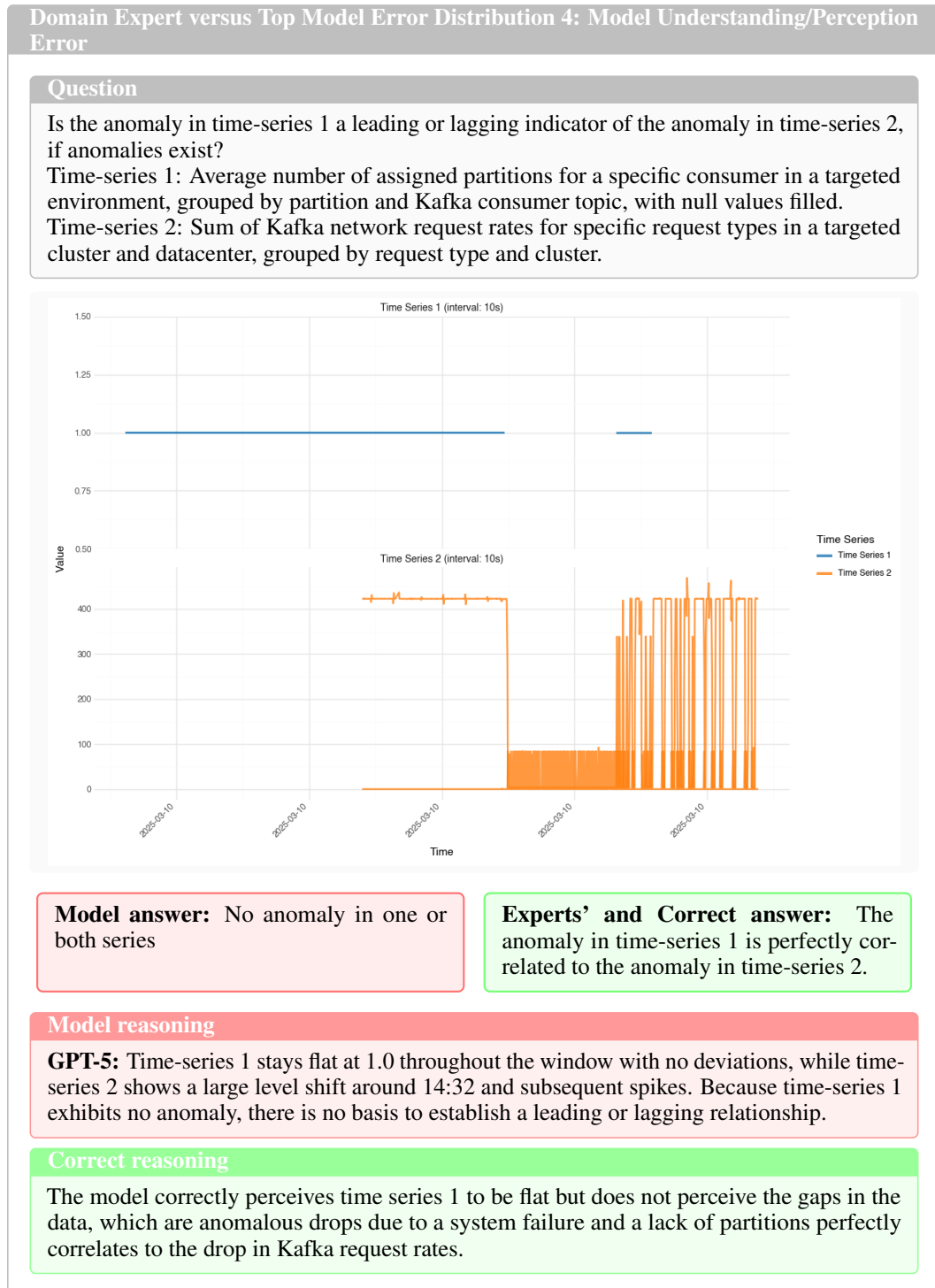


Figure 15: The model reasons incorrectly, likely due to a lack of contextualization with the time series description.

K PROMPTS

In this section we include the prompts used in benchmark choice generation and evaluation.

K.1 TIME SERIES SUMMARIZATION PROMPT

This prompt is used to summarize query strings, or unique database identifiers, to the time series in each incident. We have removed specific sensitive examples or identifiable information in this prompt.

Task Description

You are performing a caption generation task for XXXXXX queries. Your goal is to generate clear, concise, and general captions that describe the intent of a given query without leaking sensitive information about XXXXXX's architecture, services, environments, or regions.

Query Format

A standard XXXXXX subquery follows this format:

```
"operation:metric_name{{filters}}by{{group_by_tags}}"
```

Queries can consist of a single subquery (simple case) or multiple subqueries combined (e.g., ratios, differences, aggregations).

Handling Service-Specific Metrics

Some queries may contain service-specific metric names referring to internal XXXXXX services. These should be generalized in the caption. Below are some examples of such services and how to abstract them:

...

When encountering such service-specific terms in a query, do NOT mention the service name.

Instead, describe its function in a generic way.

Use 'an' instead of 'the' when referring to them.

Caption Generation Guidelines

Summarize the query's purpose in a clear and structured way.

- Describe what is being measured (e.g., memory usage, CPU load, request count) and how it is processed (e.g., ratio, sum, average).

- DO NOT exposing specific service names, regions, namespaces, env type (e.g. dev, staging) or internal identifiers.

Instead, refer to broader categories like "selected workloads", "target environment", or "specific platform".

- Include filtering and grouping context

(e.g., "excluding certain workloads" instead of listing exact exclusions).

- Adjust filtering and grouping precision dynamically:

If there are few tags, list them explicitly, otherwise use a more general description

- Maintain clarity and conciseness, ensuring captions are useful for engineers without unnecessary details.

- Follow the output format strictly.

Instructions

1. Read the query carefully

2. Follow the Caption Generation Guidelines

3. Generate a caption that accurately summarizes the query while preserving privacy

Inputs

Query: "{query}"

Output

The output should be formatted as a JSON instance that

conforms to the JSON schema below.

Here is the output schema:

```
{
  "query": "<query1> (REQUIRED)",
  "query_id": "<query_id1> (REQUIRED)",
  "caption": "<caption1> (REQUIRED)",
}
```

Examples

...

K.2 BENCHMARK OPTIONS GENERATION PROMPT

In the prompt provided below, we replace the real incident with an abridged fabricated incident in order to prevent sharing sensitive data. We also include only the first three few-shot examples provided in the prompt for brevity.

Task Description

You are performing a question-answering option generation and answer task for a given set of questions and answers.

Your goal is to generate up to 8 options for each question (including the correct answer), and label the correct answer.

Question Format

Each question either has a single query, or two paired queries.

Question Categories

There are 8 categories of questions:

1. Anomaly Presence

The anomaly presence question is a yes/no question that asks whether an anomaly is present in the time-series given.

An anomaly is present if the time-series has a value that is significantly different from the counterfactual values.

Do not generate options for this question, only identify the correct answer.

2. Anomaly Identification

The anomaly identification question asks the user to identify the channel of the anomaly in the time-series data, if an anomaly exists.

Use the options given in Existing options to generate the answer choices by including an answer choice for all single options, pairs of options, and triples of options.

DO NOT use other channels other than the existing options. Generate exactly 8 options (7 from the existing options and 1 "No Anomaly" option).

Include a "No Anomaly" option in the answer choices.

3. Anomaly Start

The anomaly start question asks the user to identify the start time of the anomaly in the time-series data, if an anomaly exists.

Do not include a timestamp that falls outside of the snapshot time range.

If the anomaly is ongoing since the start of the snapshot, the correct answer should be "Before the earliest timestamp".

The answer choices should be consistent with the time-scale of the snapshot.

Use only the snapshot_png_url and the given time range to generate the answer choices, not the incident data.

Generate 6 options for this question.

The answer choices should only consist of datetime strings.

Include a "Before the earliest timestamp" option

and a "No Anomaly" option in the answer choices.

4. Anomaly End

The anomaly end question asks the user to identify the end time of the anomaly in the time-series data, if an anomaly exists. Do not include a timestamp that falls outside of the snapshot time range. If the anomaly is ongoing at the end of the snapshot, the correct answer should be "Not resolved".

Use only the `snapshot_png_url` and the given time range to generate

the answer choices, not the incident data. Generate 6 options for this question.

The answer choices should only consist of datetime strings.

Include a "Not resolved" option and a "No Anomaly" option in the answer choices.

5. Anomaly Magnitude

The anomaly magnitude question asks the user to identify the magnitude of the anomaly in the time-series data, if an anomaly exists.

The magnitude is the maximum ratio of the anomaly values to the counterfactual non-anomalous values. Here, the magnitudes for the answer choices should be on a logarithmic scale, in the base that is most natural for the data.

If the counterfactual values are 0, use the absolute deviation from the mean counterfactual values. Generate 8 options for this question.

Include a "No Anomaly" option in the answer choices.

The answer choices should be unique and distinct from each other.

6. Anomaly Categorization

The anomaly categorization question asks the user to identify the category of the anomaly in the time-series data, if an anomaly exists.

There are 6 categories:

- Level Shift. This is when the time-series has a major sustained change in mean value compared to its counterfactual values.
- Transient Spike. This is when the time-series has a sudden spike in value, but the value returns to the normal range after a very short period of time with no intervention.
- Change in Seasonality. This is when the time-series has a change in the seasonal pattern of the data.
- Change in Variance. This is when the time-series has a major sustained change in the variance of the data.
- Change in Trend. This is when the time-series has a major change in the trend (long term increase or decrease).
- No Anomaly

Generate 6 options for this question, with the correct answer being one of the categories.

7. Anomaly Correlation

The anomaly correlation question is a paired query question that asks the user to identify whether the anomalies in two time-series are correlated.

Use the text from the incident data and the timing of the anomalies within the snapshot and incident timeline to identify the correct answer.

There is no need to generate options for this question, just to identify the correct answer.

8. Anomaly Indicator

The anomaly indicator question is a paired query question that asks the user to identify whether some anomaly in the first time-series is a leading or lagging indicator of the anomaly in the second time-series. Use the text and timestamps from the incident data as well as the order and timing of the anomalies in the snapshots to identify the correct answer. There is no need to generate options for this question, just to identify the correct answer.

Instructions

1. Read the question and context carefully
2. Use the context to generate up to 8 options for the question.
3. The options should be plausible and realistic, and should be based on the context
4. The correct option should be one of the options
5. The options should be unique and distinct from each other

Inputs

Question: <question>

This will be a question from the question categories above.

Formula: <formula>

This will be the internal company formula for the time-series data. Utilize the keywords in the formula to understand what the general behavior of the time-series should be.

Snapshot url: <snapshot_url>

This will be the URL to the snapshot image of the time-series data. Within the snapshot, the x-axis represents the time, which can vary in time-scale (e.g. minutes, hours, days, etc.). The y-axis represents the value of the time-series, which is labeled with a specific value on the far left side.

Incident data: <incident_data>

This will be the incident data for the incident that the question is about. The incident data is a markdown formatted string that has timestamped messages from the incident, which message text that is relevant to the snapshot and the time-series. The text of the message often informs whether the time-series is anomalous, its severity. The times at which different time-series are sent are often relevant to determining whether they are correlated or lagging or leading

Existing options: <existing_options>

This will be a list of existing options for the question.

Time range: <from_ts> to <to_ts>

This will be the time range of the snapshot, in a pandas datetime format. Use this time range to generate the answer choices for anomaly start and end question

The response MUST be a JSON in this format.

Respond ONLY with the JSON. All options should be strings.

Output format:

```
{
  "options": ["<option_A>", "<option_B>", "<option_C>",
             "<option_D>", "<option_E>", "<option_F>", "<option_G>", "<option_H>"],
  "correct_option": "<correct_option>"
}
```

```

## Examples
Incident data for the following questions:
# Incident XXXXX:
## High-level details
# **Summary:**
# **Root cause: Unknown**
# **Severity: SEV-X**
# **Detected at: 03-01-2025 XX:XX:XX**
# **Customer Impact Description:**
# **Customer Impact Start:**
# **Customer Impact End:**
## Timeline
# >**[Person A at 2025-03-01]:**
# Some pods are OOMing <time series 1>
# >**[Person B at 2025-03-01]:**
Seeing whether any recent deployments might have to do with this.
# >**[Person C at 2025-03-01]:**
Trying to scale.
...
# >**[Person A at 2025-03-01]:**
Ok, here's the PR.
# > **[Incident state update at 2025-03-01]:**
Incident status changed from stable to resolved.

```

Question: In the following time-series, does the time-series exhibit an anomaly in the given time range?
Time-series: Sum of unavailable replicas for Kubernetes deployments in a specific datacenter and namespace, focusing on selected applications.
Formula: <Formula>
Snapshot URL: <Snapshot URL>
Existing options: ["Yes", "No"]
Output:

```

{
  "options": ["Yes", "No"],
  "correct_option": "Yes"
}

```

Explanation: The incident timeline tells us for the given snapshot that pods are hitting resource limits with out of memory errors, which indicates an anomaly. Furthermore, within the snapshot, we see that the unavailable replicas are increasing steadily from 0 to around 30.

Question: In the following time-series, what is the start time of the anomaly, if an anomaly exists?

Time-series: Sum of unavailable replicas for Kubernetes deployments in a specific datacenter and namespace, focusing on selected applications.
Existing options: []
Output:

```

{
  "options": ["2025-03-01 10:55:00", "2025-03-01 11:05:00",
    "2025-03-01 11:40:00", "2025-03-01 11:11:40",
    "Before the earliest timestamp", "No Anomaly"],
  "correct_option": "2025-03-01 11:11:40"
}

```

Explanation: Here, we use the given time range and crucially the snapshot image's x-axis to determine the duration of the anomaly, from when it starts to rise (11:11:40) from 0 to the end of the time range

on the snapshot image (11:40:40).
The other options either do not cover the entire anomaly or are not the minimum duration between the start and end of the anomaly.

Question: In the following time-series, what is the magnitude of the anomaly?

Time-series: Sum of unavailable replicas for Kubernetes deployments in a specific datacenter and namespace, focusing on selected applications.

Existing options: []

Output:

```
{
  "options": ["0.1", "1", "5", "10", "25", "100", "125", "625"],
  "correct_option": "25"
}
```

Explanation: Here, we use the snapshot image and crucially the y-axis to determine the magnitude of the anomaly.

The maximum value is closest to 25 unavailable replicas, compared to the counterfactual non-anomalous value of 0 which occurs before the anomaly starts.

We choose powers of 5 and 10 as the scale of the data because the y-axis is between 0 and 40, and powers of 5 and 10 have many values in this range but also outside of this range that can fill the 8 answer choices.

...

K.3 PROMPT FOR FILTERING BENCHMARK QUESTIONS

Task Description

You are filtering questions and answers for a benchmark dataset based on several key criteria.

Question Format

Each question either has a single query, or two paired queries.

Question Categories

There are 8 categories of questions:

1. Anomaly Presence

The anomaly presence question is a yes/no question that asks whether an anomaly is present in the time-series given.

Key filtering criteria:

- The entire time-series should be visible in the snapshot, otherwise do not filter out any questions for this category.

2. Anomaly Identification

The anomaly identification question asks the user to identify the channel of the anomaly in the time-series data, if an anomaly exists.

Key filtering criteria:

- The entire time-series should be visible in the snapshot, otherwise do not filter out any questions for this category.

3. Anomaly Start

The anomaly start question asks the user to identify the start time of the anomaly in the time-series data, if an anomaly exists.

Key filtering criteria:

- The entire time-series should be visible in the snapshot

- The answer choices should not include any time outside of the snapshot time range
- If all answer choices are outside of the snapshot time range, then the question should be filtered out
- The answer choices should be consistent with the time-scale of the snapshot.

4. Anomaly End

The anomaly end question asks the user to identify the end time of the anomaly in the time-series data, if an anomaly exists.

Key filtering criteria:

- The entire time-series should be visible in the snapshot
- The answer choices should not include any time outside of the snapshot time range
- If all answer choices are outside of the snapshot time range, then the question should be filtered out
- The answer choices should be consistent with the time-scale of the snapshot.

5. Anomaly Magnitude

The anomaly magnitude question asks the user to identify the magnitude of the anomaly in the time-series data, if an anomaly exists. The magnitude is the maximum ratio of the anomaly values to the counterfactual non-anomalous values, or the maximum absolute deviation from the counterfactual value if the counterfactual value is 0.

Key filtering criteria:

- The answer choices should include magnitudes that are reasonable for the data. If the scale of the data is on the order of 10, but the options do not include 10, then the question should be filtered out.

6. Anomaly Categorization

The anomaly categorization question asks the user to identify the category of the anomaly in the time-series data, if an anomaly exists.

There are 6 categories:

- Level Shift
- Transient Spike
- Change in Seasonality
- Change in Trend
- Change in Variance
- No Anomaly

Key filtering criteria:

- Do not filter out any questions for this category.

7. Anomaly Correlation

The anomaly correlation question is a paired query question that asks the user to identify whether the anomalies in two time-series are correlated.

Key filtering criteria:

- If the two time-series have completely non-overlapping time ranges, then the question should be filtered out. This can be deduced from the snapshot x-axis labels, as well as the time stamps in the incident timeline. However, if the two time-series have overlapping time ranges,

then the question should not be filtered out.

8. Anomaly Indicator

The anomaly indicator question is a paired query question that asks the user to identify whether some anomaly in the first time-series is a leading or lagging indicator of the anomaly in the second time-series.

Key filtering criteria:

- If the two time-series have completely non-overlapping time ranges, then the question should be filtered out. This can be deduced from the snapshot x-axis labels, as well as the time stamps in the incident timeline.
- However, if the two time-series have overlapping time ranges, then the question should not be filtered out.

Key filtering criteria for ALL questions:

- If the snapshot does not give enough information to identify the time-series data of the specified query, then the question should be filtered out.
- If it is difficult to identify most of the x-axis labels or time-series points in the snapshot, then the question should be filtered out. This does not apply if most of the x-axis is visible but there is one point label that marks a specific time.
- If the answer options are empty, then the question should be filtered out.
- Do not filter out questions for any other reason besides the ones listed above.

Instructions

1. Read the question and context carefully
2. Use the key filtering criteria to determine if the question should be filtered out

Inputs

Question: <question>
 Formula: <formula>
 Snapshot url: <snapshot_url>
 Incident data: <incident_data>

The response MUST be a JSON in this format. Respond ONLY with the JSON. Do not include any extraneous formatting (such as newline characters or extra backslashes) or Markdown quotes. Make sure the response is an integer (either 0 or 1), 1 if the question should be filtered out, 0 if it should not be filtered out.

Output format:

```
{
  "filtered_out": <filtered_out>
}
```

Examples

Incident data for the following questions:

```
# Incident XXXXX:
## High-level details
# **Summary:**
# **Root cause: Unknown**
# **Severity: SEV-X**
# **Detected at: 03-01-2025 XX:XX:XX**
# **Customer Impact Description:**
# **Customer Impact Start:**
# **Customer Impact End:**
## Timeline
# >>>[Person A at 2025-03-01]:**
# Some pods are OOMing <time series 1>
```

```
# > **[Person B at 2025-03-01]:**
Seeing whether any recent deployments might have to do with this.
# > **[Person C at 2025-03-01]:**
Trying to scale.
...
# > **[Person A at 2025-03-01]:**
Ok, here's the PR.
# > **[Incident state update at 2025-03-01]:**
Incident status changed from stable to resolved.
Question: In the following time-series,
what is the start time of the anomaly, if an anomaly exists?
Time-series: Negative sum of application start events for
a specific node group in a production environment,
excluding certain conditions, grouped by host and datacenter.
Snapshot URL: <Snapshot URL>
Options: "['2025-03-05 03:01:00'
'2025-03-05 03:05:59'
'2025-03-05 03:07:34'
'2025-03-05 03:11:44'
'2025-03-05 03:16:20'
'Before the earliest timestamp',
'No Anomaly']"
```

Output:

```
{
  "filtered_out": 1
}
```

Explanation: The start times of all options are outside of the snapshot time range, so the question should be filtered out.

```
Question: "In the following time-series,
does the anomaly in this time-series correlate with the anomaly in the other time-series?
Time-series 1: Rate of gRPC errors in a client metrics catalog
for a specific organization, excluding canceled errors,
filtered by datacenter and canary status,
and grouped by service, error code, cause, and query source.
Time-series 2: Rate of gRPC requests to a historical data query
service without organization ID, filtered by datacenter, environment,
and specific pod, grouped by error and code."
Option: "['No, there is an anomaly in both but they are not correlated'
'Yes, there is an anomaly in both and they are correlated'
'No, there is no anomaly in either time-series'
'No, there is an anomaly only in time-series 1'
'No, there is an anomaly only in time-series 2']"
```

Output:

```
{
  "filtered_out": 1
}
```

Explanation: The two time-series have completely non-overlapping time ranges, so the question should be filtered out.

```
Question: "In the following time-series, does the anomaly in this
time-series correlate with the anomaly in the other time-series?
Time-series 1: Ratio of average CPU usage to CPU limits for a
specific processing service in a target environment,
grouped by Kubernetes deployment and cluster.
Time-series 2: Maximum value of a watermark pod autoscaler controller
```

for a specific processing service,
 filtered by datacenter and resource namespace,
 grouped by cluster and metric name, divided by 1000."
 Option: "['No, there is an anomaly in both but they are not correlated'
 'Yes, there is an anomaly in both and they are correlated'
 'No, there is no anomaly in either time-series'
 'No, there is an anomaly only in time-series 1'
 'No, there is an anomaly only in time-series 2']"

Output:
 {
 "filtered_out": 0
 }

Explanation: The two time-series have the same time range
 as we can see from the snapshot, thus the question should not be filtered out.

Question: "In the following time-series,
 which channels are exhibiting anomalies in the given time range, if any?
 Time-series: Percentage of query errors in a server query resolution system,
 calculated as the ratio of error queries to total queries,
 filtered by datacenter and grouped by method, version, gRPC method, and shard."
 Option: "'method:1' 'version:1' 'grpc_method:1' 'shard:1' 'method:1,version:1'
 'method:1,grpc_method:1' 'method:1,shard:1' 'No Anomaly']"

Output:
 {
 "filtered_out": 0
 }

Explanation: The question should not be filtered out because
 the options are valid and the time-series is visible in the snapshot.

K.4 FEW-SHOT PROMPT FOR MODEL EVALUATION

Task Description
 You are an expert at analyzing time-series data
 and answering questions about anomalies.
 Your task is to answer the given question
 about time-series anomalies by selecting the most appropriate option.
 Focus on the key aspects of the anomaly being
 analyzed and provide a clear explanation for your choice.

Input:
 - Question: <question>
 - Options: <options>
 - Snapshot PNG: URL to a PNG image of the time-series.

Question Categories
 There are 8 categories of questions:
 1. Anomaly Presence

The anomaly presence question is a yes/no question
 that asks whether an anomaly is present in the time-series given.
 An anomaly is present if the time-series has a value that
 is significantly different from the counterfactual values.

2. Anomaly Identification

The anomaly identification question asks the user to identify the channel of the anomaly in the time-series data, if an anomaly exists. You must identify the correct channels referenced in the options, and decide based on the meaning of the time series description as well as the context of the other channels to decide which channel(s) is anomalous.

3. Anomaly Start

The anomaly start question asks the user to identify the start time of the anomaly in the time-series data, if an anomaly exists. The start time is the first time the anomaly appears in the time-series. If there is no exact timestamp for the start time, the correct answer is the timestamp closest to the start of the anomaly.

4. Anomaly End

The anomaly end question asks the user to identify the end time of the anomaly in the time-series data, if an anomaly exists. The end time is the last time the anomaly appears in the time-series. If there is no exact timestamp for the end time, the correct answer is the timestamp closest to the end of the anomaly.

5. Anomaly Magnitude

The anomaly magnitude question asks the user to identify the magnitude of the anomaly in the time-series data, if an anomaly exists. The magnitude is the maximum ratio of the anomaly values to the counterfactual non-anomalous values. Here, the magnitudes for the answer choices are on a logarithmic scale, in the base that is most natural for the data. If the counterfactual values are 0, use the absolute deviation from the mean counterfactual values.

6. Anomaly Categorization

The anomaly categorization question asks the user to identify the category of the anomaly in the time-series data, if an anomaly exists. There are 6 categories:

- Level Shift. This is when the time-series has a sustained change in mean value.
- Transient Spike. This is when the time-series has a sudden spike in value, but the value returns to the normal range after a very short period of time with no intervention.
- Change in Seasonality. This is when the time-series has a change in the seasonal pattern of the data.
- Change in Variance. This is when the time-series has a major sustained change in the variance of the data.
- Change in Trend. This is when the time-series has a major change in the trend (long term increase or decrease).
- No Anomaly

Generate 4 options for this question, with the correct answer being one of the categories.

6. Anomaly Correlation

The anomaly correlation question is a paired query question that asks the user to identify whether the anomalies in two time-series are correlated. Two anomalies are correlated if they have a known

causal relation, if the time series have similar trends over time, or if they have the same underlying root causes.

7. Anomaly Indicator

The anomaly indicator question is a paired query question that asks the user to identify whether some anomaly in the first time-series is a leading or lagging indicator of the anomaly in the second time-series. Use the timing of the anomalies in the images to identify the correct answer.

Answer Format

The answer should match one of the options exactly.

Do not include the letter of the option in the answer.

Include a detailed explanation of your reasoning for the answer.

The response MUST be a JSON in this format. Respond ONLY with the JSON.

Do not include any extraneous formatting or Markdown quotes.

Output format:

```
{
  "answer": <answer>,
  "reasoning": <reasoning>
}
```