DIAGNOSING ROBOTICS SYSTEMS ISSUES WITH LARGE LANGUAGE MODELS – A CASE STUDY

Jordis Herrmann[1], **Aswath M. Gopinath**[1,2], **Mikael Norrlof**[1], **Mark Niklas Müller**[3,4]

[1] ABB Robotics [2] Linköping University [3] LogicStar.ai [4] ETH Zurich

ABSTRACT

Quickly resolving issues reported in industrial robotics applications is crucial to minimize economic impact. However, the required data analysis makes diagnosing the underlying root causes a challenging and time-consuming task, even for experts. In contrast, large language models (LLMs) excel at quickly analyzing large amounts of data. Indeed, prior work in AI-Ops demonstrates their effectiveness for IT systems. Here, we extend this work to the challenging and largely unexplored domain of robotics systems. To this end, we create SYSDIAGBENCH, an internal system diagnostics dataset for robotics, containing over 2 500 real-world issues. We leverage SYSDIAGBENCH to investigate the performance of LLMs for root cause analysis, considering a range of model sizes and adaptation techniques. Our results show that finetuned 7B-parameter models can outperform frontier models in terms of diagnostic accuracy while being significantly more cost-effective. We validate our LLM-as-a-judge results with a human expert study and find that our best model achieves approval ratings similar to our reference labels.

1 INTRODUCTION

Identifying the root cause for issues with complex industrial systems is a time-critical task but challenging and time-consuming for human experts as they struggle to analyze the large amounts of available log data. In contrast, large language models (LLMs) excel at quickly ingesting such semi-structured data. Indeed, there is substantial work in AI-Ops exploring automated diagnostics for IT systems (Díaz-de-Arcaya et al., 2024; Zhaoxue et al., 2021). However, the challenging domain of robotics systems remains largely unexplored.

This Work: Automated Diagnostics for Robotics Systems To address this challenge, and investigate the suitability of similar solutions for robotics systems, we create SYSDIAGBENCH, a dataset and benchmark for diagnosing root causes of complex robotics system failures, containing over 2 500 real-world issues. Each instance corresponds to a support ticket, containing an issue description, a set of log files, communications with the support engineers, a reference root cause extracted from expert discussions, and the ultimate issue resolution. The goal in SYSDIAGBENCH is to predict the root cause underlying the reported issue, given only the information available at the creation of the ticket.

LLM-Based Diagnostics We leverage SYSDIAGBENCH to investigate multiple LLM-based diagnostic approaches in the robotics setting, using both LLM-as-a-judge (Zheng et al., 2023) and a human expert study for evaluation. In particular, we investigate cost-effectiveness trade-offs between a range of model sizes and adaptation techniques from zero-shot prompting to full finetuning. Interestingly, we observe that even QLORA (Dettmers et al., 2023) can be sufficient to let a 7B-parameter model outperform GPT-4 in terms of diagnostic accuracy while being significantly more cost-effective. Validating our results in an expert study, we find that LLM-as-a-judge scores correlate well with human expert ratings, with our reference labels matching the experts' analysis in over half the cases and our best model achieving similar approval ratings as these reference labels.

Correspondence Author: jordis.herrmann@se.abb.com

Key Contributions Our key contributions are:

- We create SYSDIAGBENCH, a benchmark for automated root cause analysis of robotics systems, based on thousands of real-world issues (Section 3).
- We propose a range of LLM-based diagnostic tools (Section 4).
- We leverage SYSDIAGBENCH to investigate these techniques and identify the most effective and efficient strategies (Section 5).
- We validate the effectiveness of our approach using a human expert study (Section 6).

2 RELATED WORK

AI-Ops leverages machine learning (ML) in IT operations (Díaz-de-Arcaya et al., 2024) to analyze large amounts of semi-structured data such as logs and traces (Zhaoxue et al., 2021) with the goal of discovering anomalies and their root causes. As many traditional ML methods require structured data, AI-Ops long focused on developing methods enhancing (Yuan et al., 2012; Zhao et al., 2017) and parsing (He et al., 2017; Messaoudi et al., 2018) log files, using well-established methods such as SVMs (Zhang & Sivasubramaniam, 2008; Zuo et al., 2020), simple clustering techniques (Zhao et al., 2019; Lou et al., 2010), and decision tree (ensembles) (Chen et al., 2004) for the actual analysis.

LLM-based Approaches As LLMs can directly process the semi-structured log data, they have recently gained popularity in the field (Shao et al., 2022; Chen & Liao, 2022; Lee et al., 2023; Ott et al., 2021). As a representative example, Gupta et al. (2023) use an encoder architecture, pre-trained on a large amount of log data, to compute embeddings for further analysis. In contrast to these methods, we propose to directly predict root causes from log data using LLMs pretrained on internet-scale text data with limited or no finetuning.

3 SYSDIAGBENCH: A DATASET FOR ROBOTICS SYSTEM DIAGNOSTICS

SYSDIAGBENCH is an internal system diagnostics dataset focusing on root cause (RC) prediction for real-world robotics issues, constructed from a decade of industry data. Concretely, each SYSDIAGBENCH instance corresponds to a support ticket and contains a detailed problem description, a set of log files from the affected system, and a reference root cause description. Below, we first describe the information contained in a ticket and then the process of constructing SYSDIAGBENCH. Unfortunately, the underlying data cannot be published at this point due to privacy concerns.

Support Tickets A ticket is created when a reported issue cannot be resolved by the service support engineers and as a result is escalated to the product development team. Every ticket contains metadata on the affected system, e.g., the robot and application type, a detailed problem description, and a system diagnostic file capturing the system state after the issue occurred. For SYSDIAGBENCH, we consider three log file types contained in the system diagnostic that are commonly analyzed by experts when investigating a ticket. The `elog` logs all error, warning, and information events that occur during the operation of the system. This includes the exact time, error code, and additional information about the event such as relevant paths or variable names. The `print-spool` logs all outputs that are written to the console during the operation of the system. The `startup` logs all events that occur during the startup of the system, including the initialization of the system components and the loading of the software. All three log types contain a substantial amount of data at average lengths of 37k, 23k, and 8.1k tokens, respectively.

Historic Tickets which were already resolved successfully by human experts additionally contain the discussion among these experts which led to the issue's resolution, the communication with the support engineers, and the final resolution of the issue. However, even these historic tickets generally do not contain a description of the root cause (RC). Therefore, we need to extract it from the available information to obtain a reference label.

3.1 Dataset Construction

To create SysDiagBench, we collect over 12 000 historic tickets from over a decade of real-world issues and filter out those that do not contain sufficiently complete system diagnostic files with an `elog`, `pspool`, and `startup` file, leaving us with 2 585 tickets, which we split into a training, validation, and test set corresponding to 75%, 5%, and 20%, respectively.

Root Cause Extraction To extract a concise root cause description from a historic ticket to serve as a reference when scoring solutions, we leverage a strong LLM (GPT-4) (illustrated in Figure 1). Concretely, we query the LLM with the problem description, expert discussion, support engineer communication, and final resolution. We use a chain-of-thought (CoT) prompt (Wei et al., 2022), instructing the model to carefully analyze all provided information before describing the root cause (see Appendix C.1 for more details including full prompts). We highlight that the information used to create these labels is not available when a ticket is created and can thus not be used to predict the root cause at inference time. Finally, we validate the quality of the extracted root causes in a human study in Section 6.

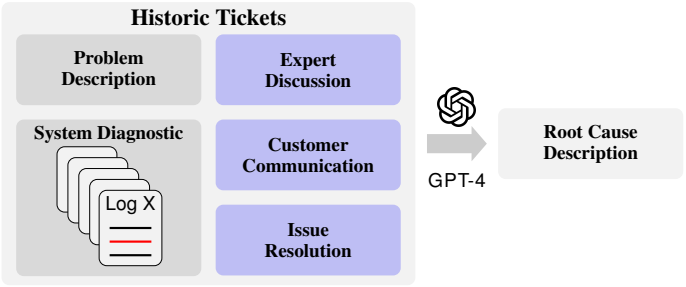

Figure 1: Visualization of the label extraction process for historic tickets based on querying a strong LLM. Note that during inference time only the grey, but not the blue, boxes are available.

3.2 Evaluation Metrics

Evaluating root cause correctness is inherently challenging, as descriptions of the same, correct root cause can be highly diverse, making similarity measures such as the ROUGE score (Lin, 2004) unsuitable. Further, there is frequently a trade-off between specificity and correctness, i.e., generic descriptions can be correct yet unhelpful, while very precise ones may get minor details wrong while still being overall very helpful. We thus adopt an LLM-as-a-judge evaluation (Zheng et al., 2023) asking a model to judge the similarity between the predicted and the reference RC on a scale of 1 to 10 (see Appendix C.3 for details). We report the mean similarity score (MSS) with respect to the reference labels as our primary evaluation metric and validate it against human experts in Section 6.

Similarity Score Callibration We repeat the above label extraction process twice more per instance, sampling at a temperature of $t = 0.5$, and compute similarity scores to the reference label obtained with greedy decoding. We thus obtain an $MSS = 7.5$, for root causes extracted by the same model with access to the same information, yielding a reference for an excellent MSS. By manual inspection, we find that a similarity score ≥ 4 typically corresponds to a correct root cause description.

4 LLM-Based Systems Diagnostic

In this Section, we describe the system diagnostic approaches we evaluate on SysDiagBench.

4.1 Preprocessing and Prompting

For both training and inference, we preprocess all log files by removing timestamps, dates, and sequence numbers. We further remove all consecutive duplicate lines and filter the `elog` to only include error and warning but not information events, as these are most likely to be relevant for diagnosing the root cause. As both the

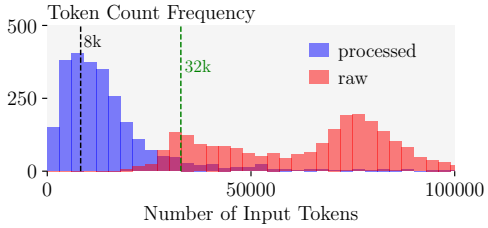

Figure 2: Token count distribution of processed finetuning inputs (blue) and corresponding raw logs (red).

`print-spool` and `startup` frequently contain tens of thousands of lines, we only retain the 10 lines before and after each error or warning event in the `elog` file. Finally, while the original `elog` file contains only integer error IDs, we map these to human-readable error descriptions using a lookup table. This preprocessing reduces the mean total token count of the log files per ticket from 68k to 16k tokens, with the distribution change illustrated in Figure 2.

The LLM input is now constructed by combining a detailed CoT instruction (Wei et al., 2022) with a context of the preprocessed log files and the issue description (see Appendix C.2 for more details). Despite our preprocessing, the resulting inputs frequently exceed 8k (68%) and even 32k (9%) tokens (see Figure 2), requiring models with long context processing capabilities.

4.2 Training for System Diagnostics

While modern LLMs have impressive zero-shot capabilities (Kojima et al., 2022), adapting them to specific tasks can improve their performance significantly (Zhao et al., 2024). However, the long input lengths make in-context learning, e.g., via few-shot prompting, unpractical for system diagnostics. We, thus, consider three adaptation techniques, full finetuning (FFT), LoRA (Hu et al., 2022), and QLoRA (Dettmers et al., 2023), with different performance-cost trade-offs (see Appendix A).

5 Experimental Evaluation

Model Selection We select models based on three criteria: i) sufficient ($\geq$ 32k tokens) context length, ii) good general reasoning capabilities, and iii) a permissive license. Based on these criteria, we choose Mixtral-8x7B (`Mixtral-8x7B-Instruct-v0.1` under Apache-2.0 License Jiang et al. 2024) and the smaller Mistral-Lite-7B (`MistralLite` under Apache-2.0 License Yin Song and Chen Wu and Eden Duthie 2023), which was specifically finetuned for long context tasks. As a reference frontier model, we consider GPT-4 (`gpt-4-32k-0613` OpenAI 2023).

Experimental Setup We use Axolotl (Axolotl, 2024) with DeepSpeed (Rajbhandari et al., 2020; Rasley et al., 2020) for finetuning with AdamW ($\beta_1 = 0.9$ and $\beta_2 = 0.95$) (Loshchilov & Hutter, 2019) on 2 to 8 NVIDIA A100s. We train for 3 epochs at an effective batch size of 64 for full finetuning and 16 for LoRA and QLoRA using an initial learning rate of 10^{-5} and a cosine decay with a warm-up ratio of 10%. Unless indicated otherwise, we use rank $r = 32$ for LoRA and QLoRA and NFloat4 + DQ (double quantization) for QLoRA.

System Diagnostic Performance We compare the performance of different models and training methods in Table 1. Interestingly, we find that the smaller Mistral-Lite-7B outperforms Mixtral-8x7B across all adaptation settings. We hypothesize this is because it was specifically trained for long context capabilities, crucial for retrieving and analyzing relevant information spread across multiple long log files. While GPT-4 is the best-performing model before adaptation, we find that our finetuned models outperform it by a significant margin, with QLoRA training yielding the best performance.

Table 1: Mean similarity score MSS for different models and adaptation methods on SysDiagBench.

Model	Base	FFT	LoRA	QLoRA
Mistral-Lite-7B	2.39	2.94	3.07	**3.27**
Mixtral-8x7B	2.25	2.69	2.24	2.30
GPT-4	**2.52**	-	-	-

Rank Deficiency as Regularization Observing that LoRA and QLoRA outperform full finetuning for Mistral-Lite-7B (see Table 1), we investigate the impact of the LoRA rank r on the model's performance, illustrating results in Figure 3. We find that while a smaller rank improves performance for LoRA, there is only a minimal effect for QLoRA. As both a reduced rank and quantization act as regularizers, we conclude that QLoRA is sufficiently regularized even at larger ranks, while LoRA benefits from the stronger regularization at lower ranks.

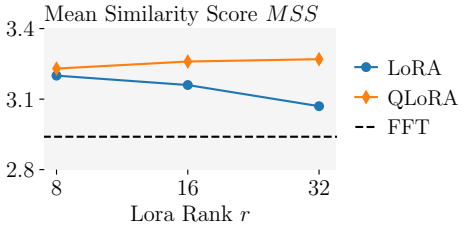

Figure 3: Mean similarity score of Mistral-Lite-7B for LoRA and QLoRA training depending on rank r.

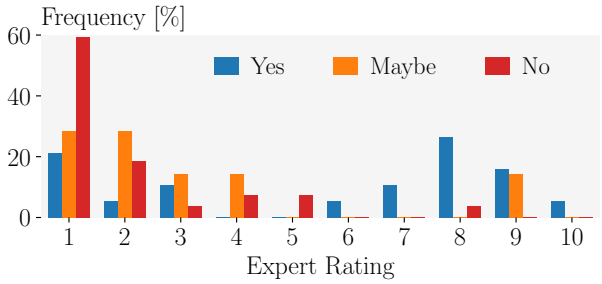

Figure 4: Human expert similarity rating between RC analysis of fully finetuned MISTRAL-LITE-7B and our reference labels depending on whether they considered the reference label to be correct (yes, maybe, or no).

Feature Importance Analysis To assess feature importance, we train MISTRAL-LITE-7B on different feature subsets using LORA and report results in Table 2. We observe that the model's performance drops significantly when excluding the `elog` and Problem Description, indicating their importance. In contrast, removing the `startup` and `printspool` improves the model's performance, suggesting that while these features may be helpful to debug compilation issues, they are less important for reported issues and can even distract the model (Jimenez et al., 2023).

Table 2: Mean similarity score of MISTRAL-LITE-7B with LORA training depending on features used.

Model	MSS
All Features	3.07
without `startup`	**3.17**
without `printspool`	3.11
without `elog`	2.85
without Problem Description	2.58
without `printspool` and `startup`	**3.22**

6 HUMAN STUDY

We conduct a study with human experts to answer the following three questions: *Q1:* Are the automatically extracted reference labels accurate? *Q2:* Does our LLM-as-a-judge evaluation correlate well with human expert ratings? And *Q3:* Are our best models helpful in root cause analysis?

Study Setup We ask 10 experts to solve a subset of tickets, leading to $n = 53$ answer sets. We provide the experts with all the tools they typically use to resolve issues and the full historic ticket data. We then ask them to describe the issue's RC and judge their confidence. Next, we let them assess our reference label, in terms of helpfulness (yes, no, maybe), correctness (on a scale from 1 to 10), and preference compared to their own answer. Where available, we let them assess their colleague's RC, in the same way. Finally, we let them rate the RCs generated by our four best models on the same scale. See Appendix E for more details.

Q1: Reference RC Quality Asked directly, whether our reference RC was correct, experts agreed (yes or maybe) in 49% of cases, saying it was as good as their own assessment in 55% of cases but only as good as their colleague's in 29%. Interestingly, they still assigned a higher or equal score to our reference RC than to their colleagues' RC in 43% of cases. Combined with experts only being highly (moderately) confident in their assessment 32% (58%) of the time, this suggests that diagnosing root causes is a particularly hard task, even for human experts with access to all findings of the original investigation. We conclude that our reference labels have high quality while not quite matching the human experts.

Q2: LLM-as-a-Judge Evaluation While overall MSS (LLM-as-a-judge) and mean expert ratings induce the same ranking, the MSS evaluated on the samples considered by the experts ranks the fully finetuned MISTRAL-LITE-7B first instead of third. Comparing per-sample LLM and expert ratings, we find a correlation of $\rho = 0.20$, which increases to $\rho = 0.34$ when considering only the samples where our reference labels were considered correct by the experts. This matches the per-sample inter-expert correlation of $\rho = 0.32$. These results again show the hardness of the root cause

analysis task with even inter-expert agreement being relatively low. However, as our LLM-as-a-judge evaluation matches inter-expert correlation, we conclude it to be a good proxy for human expert judgment where we have reliable reference labels and a decent proxy otherwise.

Q3: Model Helpfulness We compare the expert ratings of our reference and predicted RCs in Figure 5 and observe that the RCs predicted by our best two models match the reference labels extracted by GPT-4 with hindsight knowledge in half the cases. We further find that even when the *reference* labels are considered incorrect, there are instances where the *predicted* RCs are rated highly (see Figure 4). For these instances, we found by manual inspection, that our models were able to leverage a deeper understanding of the log files to identify the underlying RC. These results suggest that LLMs can provide valuable insights and help experts in diagnosing the root causes of complex robotics systems issues.

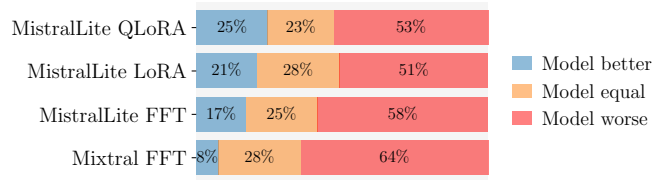

Figure 5: Frequency of human experts rating predicted RCs higher (blue), equal (orange), and lower (red) than our reference RC.

7 Conclusion

We conducted a case study on the effectiveness and cost-performance tradeoffs of large language models (LLMs) for automated root cause analysis in complex industrial settings. We considered a range of model sizes, adaptation methods, and preprocessing techniques to inform future industry applications. To this end, we created SysDiagBench, a dataset for robotics systems diagnostics, containing over 2 500 real-world issues. Our results show that finetuning even modestly sized models, especially when specialized for long contexts, can outperform frontier models like GPT-4 in terms of diagnostic accuracy while being significantly more cost-effective. We validated our results with a human expert study and found that while both our reference label extraction and LLM-as-a-judge evaluation cannot replace human experts, our best models can provide valuable insights.

8 Ethical Considerations and Broader Impact

Ethical Considerations The dataset underlying SysDiagBench contains real-world support tickets from a robotics company, which may contain sensitive information about the company's products and customers, thus precluding its public release. To mitigate the risk of privacy breaches during internal use, we have anonymized all tickets by removing all personally identifying information fields.

For our human expert study, we have recruited internal experts from the robotics company's product team, who regularly handle the support tickets constituting SysDiagBench. We have obtained informed consent from all participants and have anonymized the expert's analysis before sharing it with other experts for the inter-expert assessment. All experts were paid their regular wages during their participation in the study.

Broader Impact While we demonstrate the effectiveness of LLM-based systems for the automated diagnostics of robotics systems, we also highlight their limitations. In particular, we show that while LLMs can achieve moderately high performance and even match experts in some cases, they are unable to replace the expert's analysis for now. We thus expect that LLM-based tools will soon become useful aids for human experts, but not fully replace them in the foreseeable future, similar to many other domains.

REFERENCES

Axolotl. https://github.com/OpenAccess-AI-Collective/axolotl, accessed: 2024-03-01, 2024.

Mike Y. Chen, Alice X. Zheng, Jim Lloyd, Michael I. Jordan, and Eric A. Brewer. Failure diagnosis using decision trees. In *Proc. of ICAC*, 2004.

Song Chen and Hai Liao. Bert-log: Anomaly detection for system logs based on pre-trained language model. *Appl. Artif. Intell.*, (1), 2022.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Proc. of NeurIPS*, 2023.

Josu Díaz-de-Arcaya, Ana I. Torre-Bastida, Gorka Zárate, Raúl Miñón, and Aitor Almeida. A joint study of the challenges, opportunities, and roadmap of mlops and aiops: A systematic survey. *ACM Comput. Surv.*, (4), 2024.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. GPTQ: accurate post-training quantization for generative pre-trained transformers. *CoRR*, abs/2210.17323, 2022.

Pranjal Gupta, Harshit Kumar, Debanjana Kar, Karan Bhukar, Pooja Aggarwal, and Prateeti Mohapatra. Learning representations on logs for aiops. In *Proc. of CLOUD*, 2023.

Pinjia He, Jieming Zhu, Zibin Zheng, and Michael R. Lyu. Drain: An online log parsing approach with fixed depth tree. In *Proc. of ICWS*, 2017.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *Proc. of ICLR*, 2022.

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mixtral of experts. *CoRR*, abs/2401.04088, 2024.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. Swe-bench: Can language models resolve real-world github issues? *CoRR*, abs/2310.06770, 2023.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proc. of ICLR*, 2015.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. In *Proc. of NeurIPS*, 2022.

Yukyung Lee, Jina Kim, and Pilsung Kang. Lanobert: System log anomaly detection based on BERT masked language model. *Appl. Soft Comput.*, 2023.

Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, 2004.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Xingyu Dang, and Song Han. AWQ: activation-aware weight quantization for LLM compression and acceleration. *CoRR*, abs/2306.00978, 2023.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *Proc. of ICLR*, 2019.

Jian-Guang Lou, Qiang Fu, Shengqi Yang, Ye Xu, and Jiang Li. Mining invariants from console logs for system problem detection. In *Proc. of USENIX*, 2010.

Salma Messaoudi, Annibale Panichella, Domenico Bianculli, Lionel C. Briand, and Raimondas Sasnauskas. A search-based approach for accurate identification of log message formats. In *Proc. of ICPC*, 2018.

OpenAI. GPT-4 technical report. *CoRR*, abs/2303.08774, 2023.

Harold Ott, Jasmin Bogatinovski, Alexander Acker, Sasho Nedelkoski, and Odej Kao. Robust and transferable anomaly detection in log data using pre-trained language models. *CoRR*, abs/2102.11570, 2021.

Eunhyeok Park, Sungjoo Yoo, and Peter Vajda. Value-aware quantization for training and inference of neural networks. In *Proc. of ECCV*, 2018.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. Zero: memory optimizations toward training trillion parameter models. In *Proc. of SC*, 2020.

Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proc. of KDD*, 2020.

Yangyi Shao, Wenbin Zhang, Peishun Liu, Ren Huyue, Ruichun Tang, Qilin Yin, and Qi Li. Log anomaly detection method based on bert model optimization. In *Proc. of CLOUD*. IEEE, 2022.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *Proc. of ICLR*, 2023.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Proc. of NeurIPS*, 2022.

Benfeng Xu, An Yang, Junyang Lin, Quan Wang, Chang Zhou, Yongdong Zhang, and Zhendong Mao. Expertprompting: Instructing large language models to be distinguished experts. *CoRR*, abs/2305.14688, 2023.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *Proc. of NeurIPS*, 2023.

Yin Song and Chen Wu and Eden Duthie. amazon/MistralLite, 2023.

Ding Yuan, Jing Zheng, Soyeon Park, Yuanyuan Zhou, and Stefan Savage. Improving software diagnosability via log enhancement. *ACM Trans. Comput. Syst.*, (1), 2012.

Yanyong Zhang and Anand Sivasubramaniam. Failure prediction in IBM bluegene/l event logs. In *Proc. of IPDPS*, 2008.

Justin Zhao, Timothy Wang, Wael Abid, Geoffrey Angus, Arnav Garg, Jeffery Kinnison, Alex Sherstinsky, Piero Molino, Travis Addair, and Devvret Rishi. Lora land: 310 fine-tuned llms that rival gpt-4, a technical report. *ArXiv preprint*, abs/2405.00732, 2024.

Xu Zhao, Kirk Rodrigues, Yu Luo, Michael Stumm, Ding Yuan, and Yuanyuan Zhou. Log20: Fully automated optimal placement of log printing statements under specified overhead threshold. In *Proc. of SOSP*, 2017.

Zilong Zhao, Sophie Cerf, Robert Birke, Bogdan Robu, Sara Bouchenak, Sonia Ben Mokhtar, and Lydia Y. Chen. Robust anomaly detection on unreliable data. In *Proc. of DSN*, 2019.

Jiang Zhaoxue, Tong Li, Zhenguo Zhang, Jingguo Ge, Junling You, and Liangxiong Li. A survey on log research of aiops: Methods and trends. *Mob. Networks Appl.*, (6), 2021.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena. In *Proc. of NeurIPS*, 2023.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V. Le, and Ed H. Chi. Least-to-most prompting enables complex reasoning in large language models. In *Proc. of ICLR*, 2023.

Yuan Zuo, Yulei Wu, Geyong Min, Chengqiang Huang, and Ke Pei. An intelligent anomaly detection scheme for micro-services architectures with temporal and spatial data analysis. *IEEE Trans. Cogn. Commun. Netw.*, (2), 2020.

A BACKGORUND

Prompting Once the remarkable zero-shot capabilities of LLMs had been demonstrated (Kojima et al., 2022), a wide range of prompting schemes was proposed that aim to elicit higher quality answers from the same model by evoking a (more thorough) reasoning process (Wang et al., 2023; Yao et al., 2023; Xu et al., 2023; Zhou et al., 2023). In particular, Chain-of-Thought (CoT) prompting (Wei et al., 2022) instructs the model to "think step-by-step" when answering, which has been shown to improve performance on a wide range of tasks, with multiple follow-up works trading-off increased inference cost and better performance (Wang et al., 2023; Yao et al., 2023).

(Full) Finetuning If zero-shot performance is unsatisfactory and labeled training data is available, one can continue training the model on the specific task at hand, a process known as finetuning. In particular, full finetuning refers to training the entire model on the new task.

LoRA However, the huge size of modern LLMs makes GPU memory a bottleneck for training, with common optimizers like Adam (Kingma & Ba, 2015) and AdamW (Loshchilov & Hutter, 2019) requiring three full precision values (the gradient, and its first and second moment) to be tracked for every parameter. To alleviate this issue, LoRA (Hu et al., 2022) proposes that instead of updating all parameters in a weight matrix $\boldsymbol{W} \in \mathbb{R}^{n \times n}$ one only computes a low-rank update $\boldsymbol{AB}$ where $\boldsymbol{A} \in \mathbb{R}^{n \times k}$ and $\boldsymbol{B} \in \mathbb{R}^{k \times n}$ with $k \ll n$. We thus obtain the updated weight matrix as $\boldsymbol{W}' = \boldsymbol{W} + \boldsymbol{AB}$ and reduce the memory footprint of the optimizer from $O(n^2)$ to $O(nk)$. Finally, recent work has shown that LoRA can also be seen as a form of regularization that reduces forgetting and can thereby actually improve performance (Jimenez et al., 2023).

Model Quantization and QLoRA To reduce a model's memory footprint not only during training but also during inference, model quantization techniques have been proposed that reduce the precision of the model's weights and sometimes activations. In particular, representing the model's weight matrices using 4-, 3-, or even 2-bit precision rather than the standard (for LLMs) 16-bit half-precision representation can lead to significant memory savings (Park et al., 2018; Frantar et al., 2022; Lin et al., 2023). However, quantization can also lead to a significant drop in performance, especially if applied after training. To mitigate this issue, QLoRA (Dettmers et al., 2023) proposes to quantize the weight matrices already during training, allowing the half-precision LoRA adapters to learn to correct for the quantization errors, while still significantly reducing the memory footprint compared to standard LoRA.

B LIMITATIONS

As no ground truth root cause annotations exist for (historic) tickets, we generated reference labels for SysDiagBench using a strong LLM to extract them from the rich data available for historic tickets. While the human expert study shows the generated labels to be as good as an expert's analysis in over half the cases, they are not perfect. In particular, they are only considered correct (yes or maybe) 49% of the time by the experts. Using multiple experts to annotate the same tickets could have improved the quality of the reference labels and thus both the performance of our finetuned models and the evaluation quality, but this was not feasible due to the significant effort required to annotate tickets and time constraints of available experts.

Further, even for correct reference labels, the LLM-as-a-judge evaluation is not perfect. While we achieve a high correlation of $\rho = 0.57$ between the MSS and mean expert ratings when only considering samples with correct (yes or maybe) reference labels, the per-sample correlation remains at a moderate $\rho = 0.20$. However, even the inter-expert correlation of $\rho = 0.77$ for mean scores and $\rho = 0.32$ for per-sample correlation, remains far from perfect, highlighting again the difficulty of accurately assessing root causes. We conclude that while the LLM-as-a-judge evaluation is a valuable tool for comparing different models, it cannot substitute expert human judgment, especially for sample-level comparison. We note that for reliable sample-level comparison, a panel of experts would be needed.

Finally, we only consider two base models and a limited number of hyperparameters for our experiments due to both budget and time constraints. While we find for Mistral-Lite-7B that LoRA

```
An issue with an industrial grade robot is reported in the context
below.  What is the root cause of the reported issue?

Let's think step by step to answer this question.  First, analyze each
section in the context and systematically identify root causes and
their relative probability.  Remember that a section can have multiple
root causes or no root causes at all.  Finally, pick the root cause
with the highest relative probability and respond with the root cause
in JSON format with key as "Root Cause".  If the root cause is unknown,
respond "Unknown root cause" in JSON format.

Context:

Section for conversation between customer representative and engineers:
<CUSTOMER COMMUNICATION>

Section for conversation between engineers:
<EXPERT DISCUSSION>

Section for analysis of issue:
<ISSUE RESOLTUION>
```

Figure 6: CoT prompt used for root cause extraction, where <PLACEHOLDERS> for data from the ticket are marked red.

and QLoRA finetuning perform exceptionally well, even outperforming full finetuning, this was not the case for Mixtral-8x7B. While we hypothesize that this is due to Mistral-Lite-7B's training specifically for long context retrieval tasks, a detailed study of this effect is out of scope here and left for future work.

C Detailed Prompt descriptions

C.1 Root Cause Extraction

As described in Section 3.1, we extract the root cause (RC) from the historic tickets using a strong LLM (GPT-4) by concatenating the problem description, expert discussion, customer communication, and the final resolution with a chain-of-thought (CoT) prompt (Wei et al., 2022) instructing the model to carefully analyze all provided information before generating a root cause description. We show the full prompt used to this end in Figure 6.

C.2 Root Cuase Prediction

As described in Section 4, we use a range of LLMs to predict the root causes of the tickets in SysDiagBench. To this end, we use zero-shot prompting with the problem description and our preprocessed logs (see Section 4). We show the full prompt used for this task in Figure 7.

C.3 LLM-as-a-Judge: Similarity Score Prediction

To assess the quality of generated root cause descriptions, we use a LLM-as-a-judge evaluation procedure (Zheng et al., 2023) where we ask a model to judge the similarity between the predicted and the reference root cause descriptions on a scale of 1 to 10. We show the full prompt used for this task in Figure 8.

D Computational Requirements

Computational Requirements We provide an overview of the runtimes required for our different training setups in Table 3.

```
instruction: An issue with an industrial grade robot is reported in
the input. Determine the root cause for the reported issue.

input:
Elog log message:
<ELOG>

Error description:
<PROBLEM DESCRIPTION>

Startup log message:
<STARTUP>

Print spool log message:
<PRINT SPOOL>

output: <REFERENCE ROOT CAUSE>
```

Figure 7: Prompt used for training and inference, where <PLACEHOLDERS> for data from the ticket are marked red and the <REFERENCE ROOT CAUSE> is only provided during training. Depending on the model, we use the appropriate instruction template for both training and inference.

```
Please act as an impartial judge and evaluate the similarity of the
two analyses provided below by two different AI assistants. Both were
given the same data related to an issue in a robotics system and asked
to identify the root cause. Your job is to evaluate and quantify the
similarity between the two answers. Begin your evaluation by comparing
the two answers and identifying key differences. Do not allow the
length of the responses to influence your evaluation. Do not favor
certain names of the assistant. Be as objective as possible. After
providing your explanation, please rate the similarity on a scale of
1 to 10 by strictly following this format: "[[rating]]", for example:
"Rating: [[5]]".

[The Start of Analysis A]
<PREDICTED ROOT CAUSE>
[The End of Analysis A]

[The Start of Analysis B]
<REFERENCE ROOT CAUSE>
[The End of Analysis B]
```

Figure 8: Prompt used for LLM-as-a-judge evaluation, where <PLACEHOLDERS> are replaced with the issue specific data.

E HUMAN EXPERT STUDY

We illustrate the survey interface including all questions in Figure 9. Unfortunately, we cannot provide examples of the analyzed data and identified RCs due to the proprietary nature of the data.

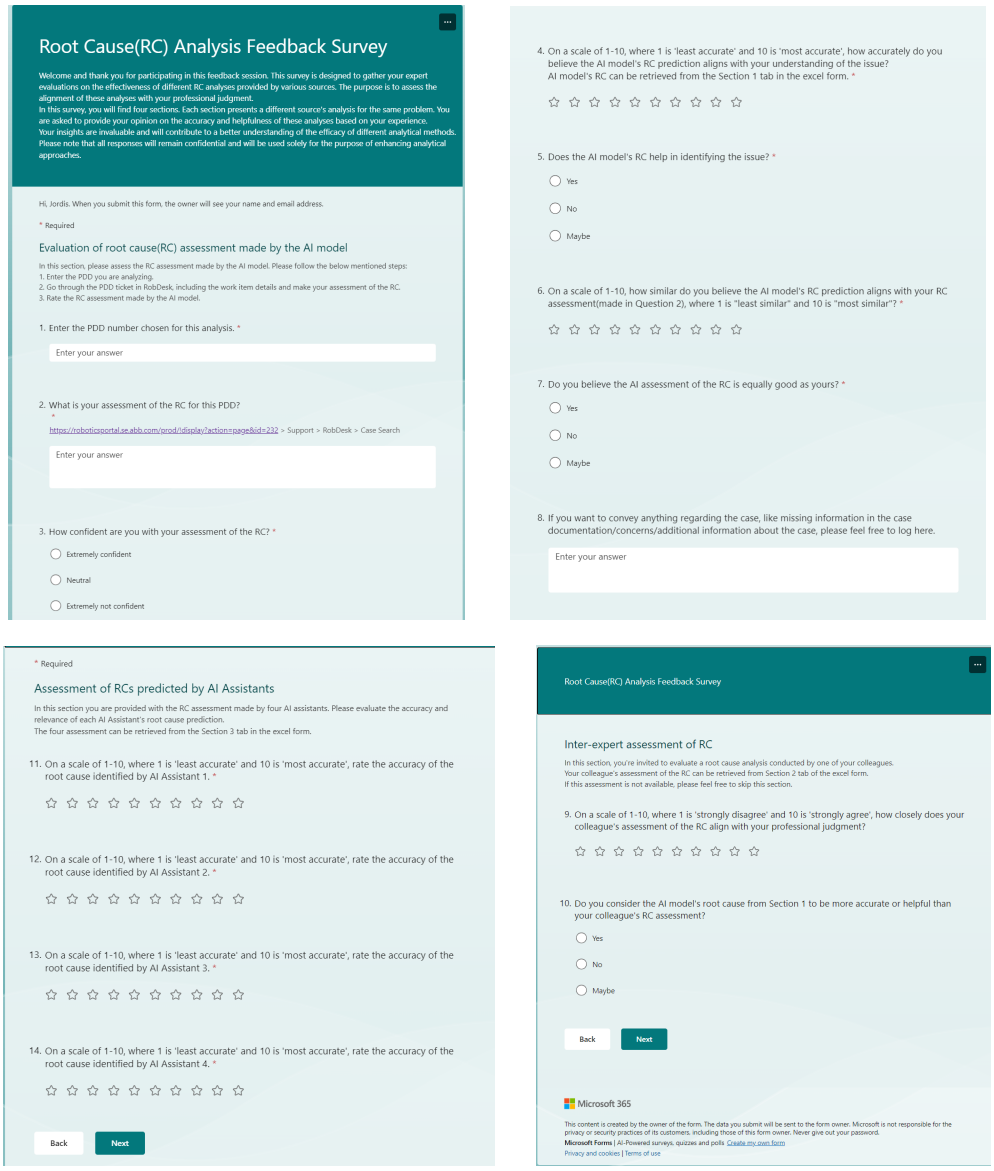

Figure 9: Human expert study questions and interface.

Table 3: GPU (NVIDIA A100) hours required for different training setups.

Model	Training Mode	Training Time [h]
MISTRAL-LITE-7B	FFT	23
	LoRA	20
	QLoRA	20
MIXTRAL-8X7B	FFT	64
	LoRA	40
	QLoRA	40