From Language to Fault Detection: Small Language Models for Optical Network Automation

Andrei N. Ribeiro*, Raul P. C. Costa*, Fabrício R. Lobato*, Moisés F. Silva†, Carlos Natalino‡, Ingrid O. Ramos*, João C. W. A. Costa*

*Federal University of Pará, Belém, Brazil
{andrei.ribeiro, raul.costa, ingrid.ramos}@itec.ufpa.br, {frl, jweyl}@ufpa.br
†Los Alamos National Laboratory, Los Alamos, USA
mfelipe@lanl.gov
‡Department of Electrical Engineering, Chalmers University of Technology, Gothenburg, Sweden
carlos.natalino@chalmers.se

***Abstract*—Optical networks are increasingly complex and mission-critical, making timely, accurate fault detection essential to maintaining service reliability. Traditional threshold-based approaches struggle with heterogeneous traffic patterns and evolving failure modes, motivating the adoption of more flexible, data-driven solutions. In this work, we investigate the use of compact Small Language Models (SLMs) for optical network automation and propose a local SLM-based AI agent that automates fault detection by interpreting natural-language requests and executing PCA-based analytics with Qwen2.5-7B. The agent leverages retrieval-augmented generation on curated optical networking and ML documentation to translate operator intent into executable analytics workflows, running entirely on local infrastructure. To the best of our knowledge, this is the first demonstration of SLM-driven fault-detection automation in optical networks, showing that compact language models can reliably implement PCA-based monitoring with performance comparable to a reference implementation.**

***Index Terms*—optical networks, small language models, fault detection automation.**

I. Introduction

Optical networks form the backbone of modern digital infrastructure, supporting data-intensive services that demand high reliability and minimal outage. As these systems scale in size and complexity, traditional threshold-based methods struggle to ensure timely and accurate fault management. Machine learning (ML) techniques have shown promise in this context, offering data-driven insights and adaptability that outperform conventional approaches [1]. As network automation advances, systems must make accurate decisions while interacting naturally with network administrators. Natural-language interfaces simplify the adoption, deployment, and operation of intelligent management tools.

Large language models (LLMs) represent a further step toward such automation, enabling natural-language-driven control and flexible integration with network management tasks, allowing operators to interact with systems without explicit programming and thereby reducing the barriers to intelligent system adoption. A few works leveraging LLMs for optical network automation have emerged in recent years, none of which focus on adopting Small Language Models (SLMs) [2]. These models retain the key advantages of LLMs while offering lower computational requirements and better adaptability to specialized scenarios such as optical networks [3]. When integrated with retrieval-augmented generation (RAG) techniques, SLMs can leverage domain-specific knowledge from relevant data sources, enhancing accuracy and reducing hallucinations without extensive retraining [3].

Building on these ideas, this paper presents an SLM-based local AI agent to automate fault detection in optical networks and evaluates the feasibility of compact language models for streamlining the implementation of intelligent monitoring and control solutions. This paper presents the first demonstration of this concept for fault detection in optical networks. Domain expertise is provided through RAG using a curated documentation of optical networking/ML literature, enabling the model to translate natural-language prompts into executable analytics workflows. In this study, we evaluate the use of a lightweight SLM, Qwen2.5-Coder 7B [3], to automate fault detection by having the model implement a principal component analysis (PCA)–based detection method and compare its performance against a reference PCA implementation. Compared to frontier LLMs such as GPT-4 class models, which exceed 100 billion parameters and require cloud-based inference, this SLM operates with approximately 7 billion parameters and fits within 8 GB of GPU memory, enabling fully local execution.

II. Proposed SLM-Based AI Agent

The proposed SLM-based agent is implemented following the steps depicted in Figure 1. *Step 1* begins with the user submitting a request and uploading the required datasets and optical-domain-specific documentation. These inputs are stored in designated system folders for organized data management, ensuring that all necessary information is readily available for subsequent processing steps. In *Step 2*, the available documentation is split into manageable chunks and embedded with the all-MiniLM-L6-v2 model, enabling efficient semantic matching for information retrieval.*Step 3* stores all embeddings in a vector database, forming the knowledge base for future queries. This centralized storage mechanism serves as the

978-3-903176-78-2

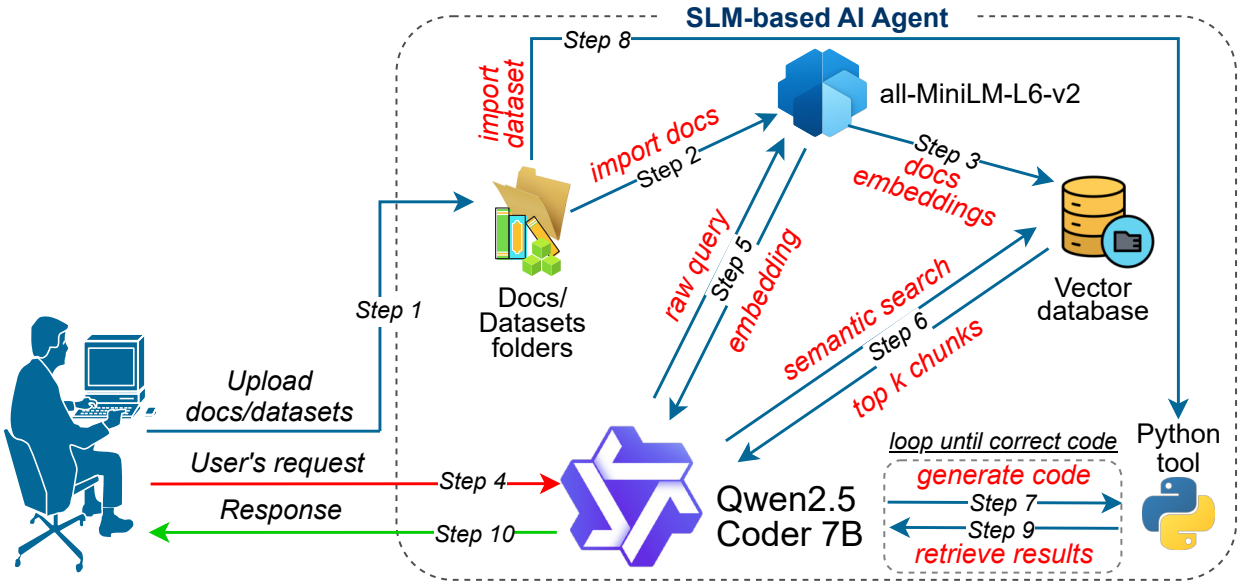

Fig. 1. The complete workflow of the proposed SLM-based AI agent.

backbone of the system's retrieval capabilities, enabling efficient handling of large volumes of domain-specific data. Next, in *Step 4*, the user inserts their query into the SLM, which is embedded in *Step 5* for intent and context matching in the vector database. *Step 6* executes the RAG process, where the query embeddings are compared with the stored embeddings using cosine similarity to retrieve the *top_k* relevant chunks. These chunks, along with the query, are passed to the SLM for further reasoning. This RAG process provides contextually relevant knowledge, enabling the system's ability to tackle complex domain-specific queries.

From Fig. 1, when the SLM receives the query, it first analyzes its nature. If the query does not pertain to coding or ML implementation, the system proceeds directly to *Step 10* and delivers the response to the user. However, for requests involving general coding, *Step 7* first invokes the Python tool for code execution. In *Step 8*, the Python tool imports the dataset specified in the query, ensuring the code execution is contextually aligned with the user's request. *Step 9* captures the execution result and passes it back to the SLM. Coding errors are addressed by a correction loop comprising *Steps 6-8*. This iterative loop allows the SLM to analyze errors, rethink the solution, and generate a new code until a successful execution is achieved. Once a corrected output is obtained, the system concludes the workflow with *Step 10*, where the SLM provides the final response to the user.

III. Experimental Setup and Results Discussion

To evaluate the proposed SLM-based agent, we conducted experiments using two datasets (CSV files) derived from the hard-failure dataset available in [4]. The training dataset (*nonfaildata.csv*) has 10884 samples from normal conditions, and the testing dataset (*faildata.csv*) has 2353 samples from normal conditions and 711 samples from fault conditions. Both datasets comprise five timestamp-related features and four EDFA input powers. The testing file also consists of a "Failure" label for evaluation purposes. Additionally, a *README.md* file containing specific information from the optical testbed is collected from [4] for RAG evaluation purposes. Also, an ML and Python-related book [5] was uploaded for RAG, enriching the SLM's reasoning capabilities. The experimental setup utilized an AMD Ryzen 7 5800H processor with Radeon Graphics (3.20 GHz), 16 GB RAM, and an AMD RX 6500M GPU for local execution. The Qwen2.5-Coder 7B model was configured with a temperature of 0.1, a context window of 4096 tokens, and a token generation limit of 2048. RAG parameters included chunk sizes of 1000 tokens with 100-token overlaps, a maximum context length of 2000 tokens, and a similarity threshold of 0.6 for retrieving the top 10 chunks. The number of maximum code correction iterations is set to 10. The manual PCA implementation, serving as the baseline for evaluating the SLM-based agent, involves training a 1-component PCA model exclusively on normal-condition samples, a standard practice in unsupervised anomaly detection, in which the model learns a compact representation of normal behavior and flags deviations as faults. Reconstruction errors (RE) are then computed on the test set, and the anomaly threshold is set at the 99th percentile of training REs. Metrics such as false positives (FP), false negatives (FN), accuracy, F1-score, and average RE are used for evaluation, following the methodology outlined in [6].

Fig. 2 illustrates the effectiveness of the proposed SLM-based local agent in handling two distinct user queries. Before, the user first uploads the ML book and the file containing

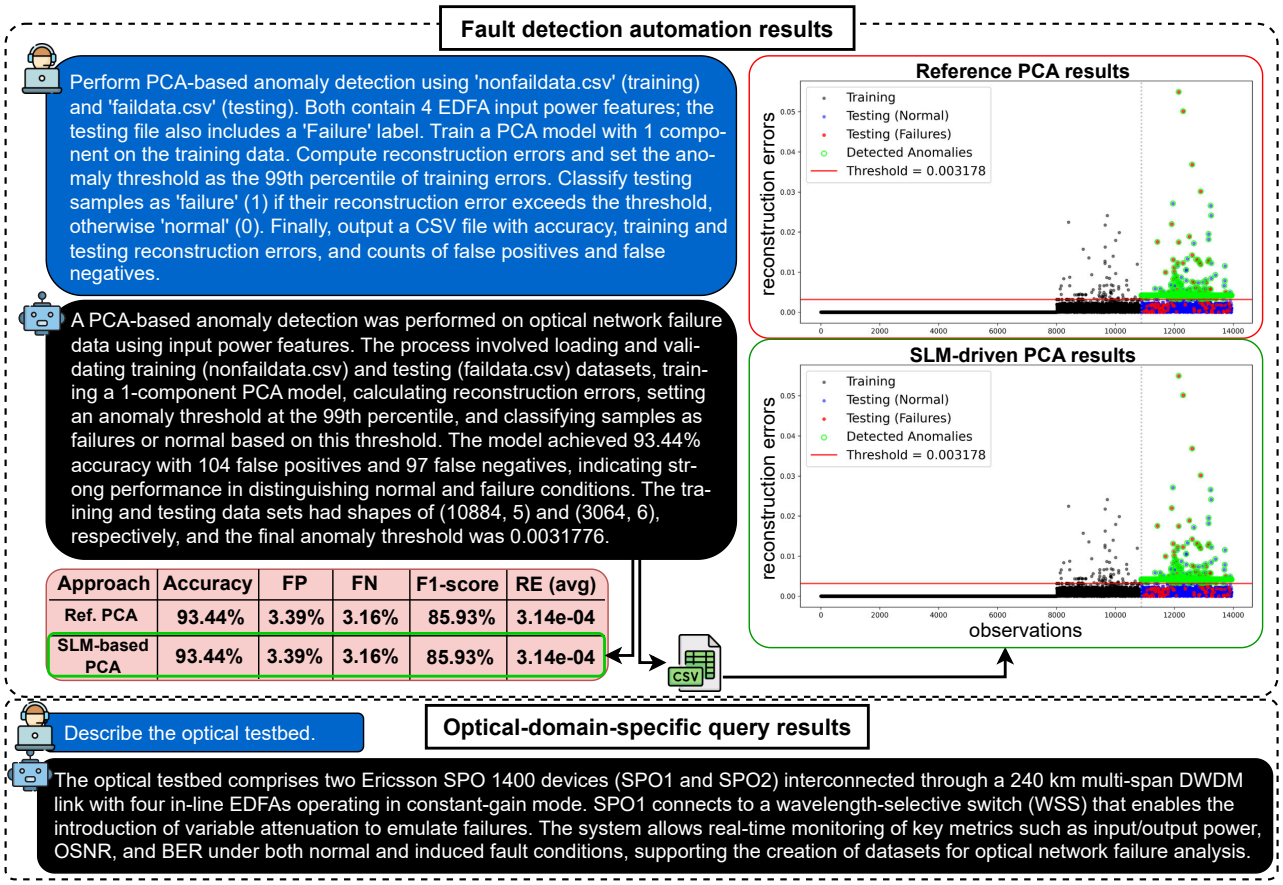

Fig. 2. Results of the SLM-based agent for fault detection automation and optical-domain queries.

specialized information about the optical testbed to be processed by the RAG. In the first query, requesting PCA-based fault detection, the SLM model correctly implemented PCA, provided reasoning on the model's performance results, and generated a CSV file containing REs for the training and testing datasets. These errors were used to produce the figure on the right, which illustrates PCA fault-detection performance by showing the REs for the training and testing sets, along with the detection threshold and the identified failures. From the table in Fig. 2, both methods achieved identical performance for the evaluation metrics. Also, notably, the SLM-driven PCA results could replicate the REs figure on the side from the manual PCA. In the second query, since the model has access to domain-specific documentation, the user was able to ask about the optical setup. The model successfully provided a comprehensive description of the optical testbed, including details of its architecture, such as the use of Ericsson SPO 1400 devices, and the system's ability to induce failures, thereby supporting the creation of optical datasets.

IV. Conclusion

In this work, we proposed an SLM-based agent that leverages the Qwen2.5-Coder 7B to effectively automate fault detection in optical networks and address domain-specific user queries via RAG. The proposed approach matched the performance of the manual PCA implementation and correctly handled optical-domain-specific queries.

Acknowledgments

This work was supported by the National Council for Scientific and Technological Development (CNPq), Brazil.

References

[1] Cruzes, Sergio. "Failure management overview in optical networks." IEEE Access (2024).
[2] D. Wang, Y. Wang, X. Jiang, Y. Zhang, Y. Pang, and M. Zhang, "When Large Language Models Meet Optical Networks: Paving the Way for Automation," Optical Express, vol. 32, no. 15, pp. 20 876–20 896, 2024.
[3] Wang, Fali, et al. "A comprehensive survey of small language models in the era of large language models: Techniques, enhancements, applications, collaboration with llms, and trustworthiness." ACM Transactions on Intelligent Systems and Technology (2024).
[4] InRete Lab, "Optical Failure Dataset". Scuola Superiore Sant'Anna, 2021. [Online]. Available: https://github.com/Network-And-Services/optical-failure-dataset.
[5] Raschka, Sebastian, and Vahid Mirjalili. "Python machine learning: Machine learning and deep learning with Python, scikit-learn, and TensorFlow 2." Packt Publishing Ltd, 2019.
[6] Sales, R. F., et al. "Disaggregated confidentiality-preserving scheme for fault detection in optical networks." 2024 Optical Fiber Communications Conference and Exhibition (OFC). IEEE, 2024.