Experimental Demonstration of Local AI-Agents for Lifecycle Management and Control Automation of Optical Networks [Invited]

CHENYU SUN[1,2,3,*], XIN YANG[1,4], NICOLA DI CICCO[4,5], REDA AYASSI[1], VENKATA VIRAJIT GARBHAPU[1], PHOTIOS A. STAVROU[2], MASSIMO TORNATORE[4], GABRIEL CHARLET[1], AND YVAN POINTURIER[1]

[1] *Optical Communication Technology Lab, Paris Research Center, Huawei Technologies France SASU, Boulogne-Billancourt, France*
[2] *Communication Systems Department, EURECOM, Sophia Antipolis, France*
[3] *Sorbonne Université, Paris, France*
[4] *Politecnico di Milano, Milan, Italy*
[5] *OPTIT S.r.l., Bologna, Italy*
[*] *chenyu.sun1@huawei.com*

Compiled April 2, 2025

This paper presents an innovative approach to automating the full lifecycle management of optical networks using locally fine-tuned large language models (LLMs) and digital twin technologies. We experimentally demonstrate the integration of generative AI and digital twins to create powerful AI-Agents capable of handling the design, deployment, maintenance, and upgrade phases in the lifecycle of optical networks. By deploying and fine-tuning LLMs locally, our framework eliminates the need for public cloud services, thereby ensuring data privacy and security. The experimental setup includes a commercial-product-based testbed with 8 optical multiplex sections in the C-band, showcasing the effectiveness of the AI-Agents in various automation tasks, such as API-calling for service establishment and periodic power equalization, as well as log analysis for troubleshooting. The results highlight significant improvements in operational accuracy and efficiency, underscoring the feasibility of this approach in real-world scenarios. This work represents a significant advancement toward intent-based networking, showcasing the transformative potential of AI in automating and optimizing optical network operations.

http://dx.doi.org/10.1364/ao.XX.XXXXXX

1. INTRODUCTION

Artificial intelligence (AI) and digital twin (DT) technologies have been prominent and rapidly evolving topics in recent years. Among various AI techniques, large language models (LLMs), a subset of Generative AI (GenAI) that leverage advanced neural networks to comprehend and generate natural language text, have garnered significant attention for their wide-ranging applications across multiple domains. Researchers in optical communications and networking are currently exploring innovative applications of LLMs and DTs. As shown in Tab. 1, LLMs can assist in analyzing alarm logs and troubleshooting network issues [1–4] and they can automate network management tasks by generating network configurations based on application programming interfaces (APIs) [5–10]. DTs have also recently raised increasing attention in optical networks for automation and management [11–14]. A network DT can analyze physical-layer monitoring data based on physical models or machine learn-

Table 1. State-of-the-Art

Ref.	Tech.	Applications
[1, 2]	LLM	Alarm analysis using cloud LLMs.
[3]	LLM	Network assistant using cloud LLMs.
[4]	LLM	Log analysis using local LLMs.
[5–7]	LLM+DT	Intent-based network using cloud LLMs.
[8]	LLM+DT	Intent-based network using local LLMs.
[9, 10]	DT	API-based network control automation.
[11–15]	DT	Digital twin optical networks (DTONs).
[16]	DT	Failure management.

ing models (or a combination of the two [15]). Moreover, DTs can be applied to failure management in optical networks [16]. Then, the DT can be used as a sandbox to test proposed physical layer changes before implementing any operations in the physical world by first emulating such changes and predicting their impact in the DT itself.

However, previous works [1–3, 5–7] were using LLMs on the public cloud, leading to data privacy concerns. Though [8] used a local LLM for network auto-configurations, it only achieved 80% accuracy in translating human intents to the appropriate network configurations. In this paper, we demonstrate for the first time how to use fine-tuned local LLMs for automating an optical network. By deploying and fine-tuning LLMs locally, we avoid using a public cloud and achieve 100% accuracy in interpreting human requests with low hardware requirements. Moreover, we demonstrate how to build AI-Agents based on DT and other tools for automating an optical network during the four key steps of its lifecycle: design, deployment, maintenance (including power re-equalization and troubleshooting), and upgrade. After the experimental demonstration, we conclude the pros and cons of applying LLMs in optical networks based on our experience.

The rest of this paper is organized into six sections. In Section 2, we introduce the concepts of LLM and AI-Agent. In Section 3, we introduce the DT and software-defined networking for optical networks. In Section 4, we show our experimental setup and framework for locally deployed AI-Agents. In Section 5, we demonstrate the applications of AI-Agents for lifecycle control and management of optical networks. In Section 6, we discuss the pros and cons of LLMs in optical network control and management. In Section 7, we conclude our study and discuss possible future perspectives.

This paper is an extended version of our post-deadline paper (PDP) at the European Conference on Optical Communication (ECOC) in 2024 [17]. In this paper, we provide more details on the implementation of AI-Agents for optical network automation. Key novel contributions include:

- Detailed discussions on employed methodologies in Section 2 and relative implementation details in Section 4;
- More results showcasing the network lifecycle control and management in Section 5;
- Extended literature review and discussions about the pros and cons of LLM-based systems in optical network control and management in Section 6.

2. LARGE LANGUAGE MODELS AND AI-AGENTS

LLMs are a class of AI systems that employ advanced neural network architectures, such as transformers, to process, understand, and generate natural language text. These models underpin a wide range of applications, including chatbots, text summarization, language translation, and more. In particular, the Transformer architecture has revolutionized natural language processing by introducing a mechanism called self-attention, which evaluates the relative importance of words within a sentence. This mechanism enables LLMs to grasp context and relationships in text more effectively than previous methods [18].

An LLM comprises two key components: the tokenizer and the model weights stored in a safetensor file. The tokenizer converts text inputs into discrete tokens (e.g., words, subwords, or punctuation) and maps them into a high-dimensional vector space. The safetensor file encapsulates the model's learned

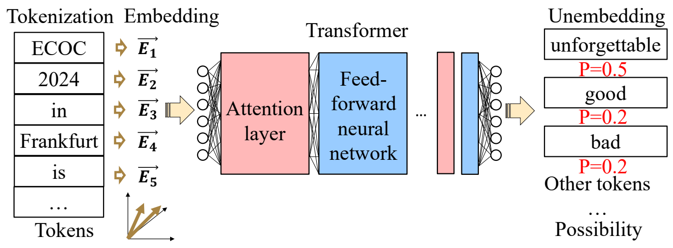


Fig. 1. Transformer-based LLM.

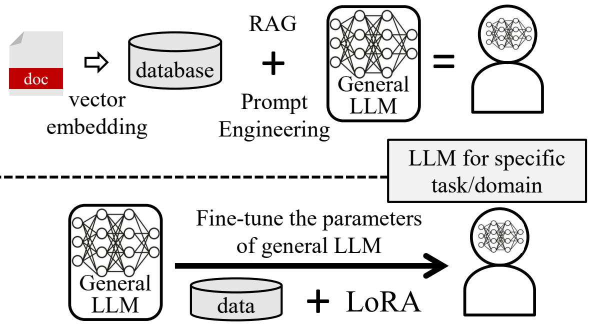


Fig. 2. Two application-specific LLMs, using one or several of: Prompt engineering, RAG, and fine-tuning.

parameters, including the weights of attention layers and feed-forward neural networks. The model processes tokens through the transformer and predicts the next token with associated probabilities, enabling natural language generation. Fig. 1 offers a simplified overview of the Transformer architecture, emphasizing the core components that are pertinent to its application in optical network management. This depiction is a basic representation, and actual Transformer implementations can vary significantly, incorporating different encoder and decoder blocks and other modifications tailored to specific large language models. This simplified version is intended to provide clarity and focus on the essential aspects relevant to the discussions in this paper.

Due to the complexity of transformers, these safetensor files can be exceptionally large, as they often include billions of parameters in high-precision formats like 32-bit floating-point (fp32). To address memory and hardware constraints, quantization methods reduce the precision of these parameters, enabling storage in formats such as 16-bit floating-point (fp16), brain floating-point (bf16), or even 8-bit integers (int8). For example, an LLM with 7B parameters in fp16 has a size of $7 \cdot 10^9 \cdot 2\ Byte = 14\ GB$, while the same model with parameters in int8 only has half size, i.e. $7\ GB$, which fits on a customer-grade laptop without a powerful graphics processing unit (GPU). However, quantization introduces a trade-off between memory efficiency and model accuracy, as reduced precision can degrade performance.

Pre-trained LLMs like OpenAI's ChatGPT [19], Meta's LLaMA [20], and Mistral AI [21] are built on general-purpose knowledge bases. While they contain foundational knowledge, their performance in specialized tasks remains limited due to the lack of domain-specific training.

A. Prompt Engineering and Retrieval-Augmented Generation

Prompt engineering is the practice of crafting effective input prompts to optimize the performance of LLMs. By carefully designing the phrasing, structure, and context of prompts, users can guide the model to generate more accurate, relevant, and context-aware outputs. This approach is particularly useful for tailoring LLM responses to specific tasks or domains, such as technical problem-solving or creative writing. Prompt engineering often involves iterative refinement, leveraging strategies like providing explicit instructions, examples, or formatting guidelines to maximize the model's understanding and utility.

As shown in Fig. 2(top), retrieval-augmented generation (RAG) is a technique that combines retrieval-based methods with generative models to enhance the accuracy and relevance of generated responses by incorporating external knowledge. The retriever searches external knowledge bases (e.g., technical documents or databases) for relevant data, which is then combined with the user query. The generator (LLM) uses this context to produce fluent, domain-specific responses. This approach ensures up-to-date , accurate, and specialized knowledge integration, especially in dynamic fields like optical networks. RAG's key benefits include improved factual accuracy, adaptability to specific domains, and reduced dependency on static pre-trained knowledge.

B. Fine-Tuning the Parameters of an LLM

Since transformers are based on neural networks, another approach for adapting LLMs to specific tasks/domains is fine-tuning the parameters of the model. As shown in Fig. 2(bottom), the process involves further training a pre-trained model on a smaller, task-specific dataset, allowing it to specialize while leveraging previously acquired general knowledge. There are various fine-tuning approaches, including full fine-tuning, which updates all model parameters, and parameter-efficient methods like low-rank adaptation (LoRA) [22], adapter tuning, and prefix tuning, which modify only a subset of parameters or introduce new trainable elements to reduce computational costs.

LoRA freezes the pre-trained model parameters and injects trainable rank decomposition matrices into each layer of the Transformer architecture, which reduces hardware requirements and allows for efficient adaptation to new tasks.

In summary, fine-tuning integrates knowledge into the model's parameters during training, while RAG retrieves and uses external knowledge on-the-fly during inference.

C. LLM-Based AI-Agents

An AI-Agent is an entity powered by AI, designed to perceive its environment, process data, make decisions, and take actions to achieve specific goals. These agents can learn, adapt, and interact with humans or systems, making them versatile tools across various fields. An LLM-based AI-Agent extends this concept by integrating an LLM with specialized tools and domain knowledge for precise, domain-specific tasks. In optical communications, AI-Agents can automate optical networks by leveraging tools like DT, which we will introduce in the next section. And we will present how to build AI-Agents for optical networks in Section 4.

3. DIGITAL TWIN AND SOFTWARE-DEFINED NETWORKING

In this section, we introduce the tools used to build AI-Agents for optical networking.

DTs, which are software replicas of physical systems, are utilized to monitor, analyze, and predict network behavior [23],

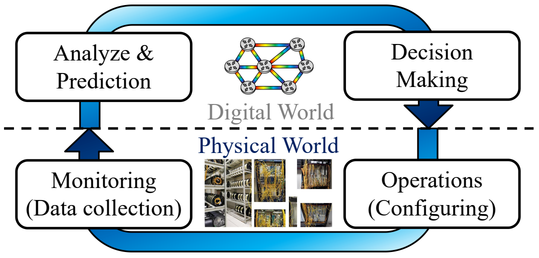


Fig. 3. DT leverages a monitoring-prediction-decision-operation control loop, promises the reliability of networks.

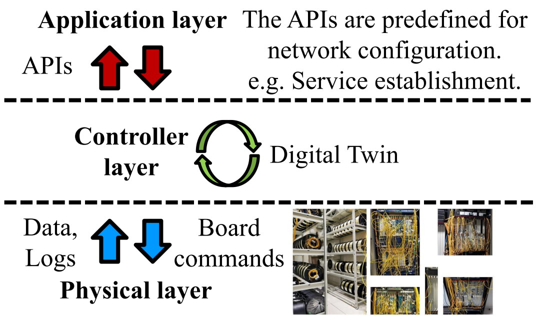


Fig. 4. The digital twin sits at the intersection of the application and physical layers, e.g., within an SDN controller.

thereby enhancing decision-making processes before implementing any physical adjustment (Fig. 3). A key role of a DT is to estimate and predict the quality of transmission (QoT), e.g., signal-to-noise ratio (SNR), optical signal-to-noise ratio (OSNR), and generalized signal-to-noise ratio (GSNR). A second key role is to optimize the SNR of optical channels in the network, for instance, by balancing the amplified spontaneous emission (ASE) and non-linear (NL) noise [24]. In addition, emulating operations and predicting the impact of these operations in the DT, thereby making better decisions before implementing any operation in the physical world.

Software-defined networking (SDN) is a networking paradigm that separates the control plane from the data plane, enabling centralized and programmable network management. SDN architecture is typically divided into three layers: the application layer, the control layer, and the physical layer, as shown in Fig. 4. The application layer consists of software applications that define network behavior and services, such as service establishment, performance analysis, and troubleshooting in optical networks. The control layer, often implemented as a centralized SDN controller, acts as the brain of the network, translating application requirements into actionable instructions, e.g., configuration of line parameters in optical networks. In addition, the controller can extract data and event logs from network elements in the physical layer. Finally, the physical layer comprises the underlying hardware, such as wavelength selective switches (WSSs), optical amplifiers, and transponders in optical networks, which execute the commands from the control layer. This layered approach enhances flexibility, scalability, and adaptability, making SDN a key enabler of modern, dynamic networks, including optical and cloud-based infrastructures.

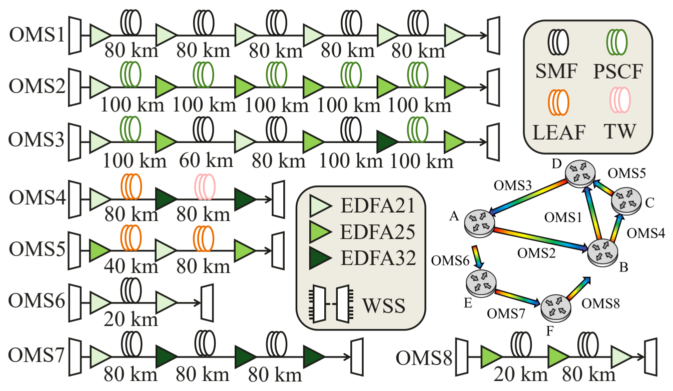

OMS1
80 km 80 km 80 km 80 km 80 km
OMS2
100 km 100 km 100 km 100 km 100 km
OMS3
100 km 60 km 80 km 100 km 100 km
OMS4
80 km 80 km
OMS5
40 km 80 km
OMS6
20 km
OMS7
80 km 80 km 80 km
OMS8
20 km 80 km
SMF PSCF
LEAF TW
EDFA21
EDFA25
EDFA32
WSS
A B C D E F
OMS1 OMS2 OMS3 OMS4 OMS5 OMS6 OMS7 OMS8

Fig. 5. Experimental setup.

4. DEPLOYMENT OF LOCAL AI-AGENTS ON THE COMMERCIAL-PRODUCT-BASED TESTBED

Our proposed framework combines prompt engineering, RAG, and fine-tuning to address diverse tasks during the lifecycle of optical networks. To handle specialized tasks effectively, we implement a multi-agent framework, where each sub-agent has a specific role. These agents collaborate and coordinate to solve complex tasks, leveraging technologies such as LLMs, retrieval systems, and network DTs.

A. Testbed

Commercial network elements and boards are used to build our C-band optical network testbed, consisting of 8 optical multiplex sections (OMSs), for a total of 25 spans and 1980 km of fiber (Fig. 5). The OMSs are heterogeneous in fiber length and type, and (model of) of erbium doped fiber amplifiers (EDFAs). The fiber types include G.652.D single mode fiber (SMF), G.654.E pure-silica-core fiber (PSCF), G.655.D large effective area fiber (LEAF), and G.655.D true-wave fiber (TW). The EDFAs shown in Fig. 5 have various tunable gain ranges, EDFA21 has a range of 16-21dB, EDFA25 has a range of 19-25dB, EDFA32 has a range of 23-32dB. The WSS grid is set to 100 GHz channel spacing within the 6 THz C-band.

Channel loading is emulated with an ASE source. A real-time 400 Gb/s (PDM-PCS16QAM) transponder is used to measure the bit error rate (BER), thereby calculating the SNR.

The testbed is automated with our SDN framework named AI-Light [25]. The SDN controller collects the data from the physical layer and implements the DT to perform various optimization algorithms (routing, spectrum allocation, power optimization, ...). The power spectra can be monitored by either an optical spectrum analyzer (OSA) or a commercial optical performance monitor. We can also emulate link failures by plugging out the fibers or disabling the optical amplifiers in the testbed, and collect the alarm logs from network elements.

B. Local Deployed LLMs

We leverage an 8-GPU (NVIDIA Tesla V100-PCIe-32GB, launched in 2018) server for locally deploying and fine-tuning the LLMs. Each GPU has 32 GB memory and delivers 14 TFLOPS (Tera floating point operations per second) of fp32 performance. The 8-GPU memory is 256 GB, hence, we can deploy and fine-tune different open-source LLMs locally, namely Mistral-7B-Instruct [26] (model size 14.5 GB) that fits on a single GPU , Mixtral-8x7B-Instruct [27] (model size 90.4 GB), which requires multiple GPUs . These open-source models are downloaded from Hugging Face [28], the biggest community of open-source LLMs. Mistral AI develops LLMs supporting multiple European languages, and we use it to work with not only English but also French, Italian, Spanish, German, Greek, etc.

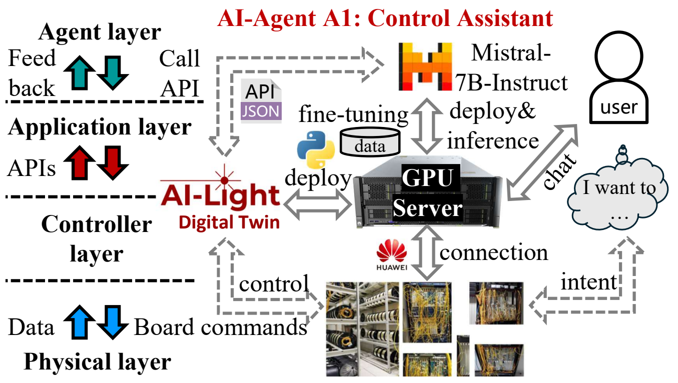


Fig. 6. AI-Agent A1 for network control based on user's intent. (Solid arrows represent direct connections, while dash arrows represent indirect connections.)

C. AI-Agents for Control and Management of Optical Network

We build two task-specific AI-Agents based on LLMs and different tools: AI-Agent A1 for network automation, which fed LLMs with API descriptions for interaction with our SDN and DT; AI-Agent A2 for network management, which uses LLMs and embeds product documentation for system design and log analysis. Using multiple specialized AI-Agents instead of a single general-purpose agent allows each agent to be optimized for specific tasks, leading to more accurate and reliable performance. This approach reduces complexity, improves scalability, and minimizes the risk of hallucinations by leveraging task-specific expertise and focused training.

C.1. API-Calling to Control Optical Networks

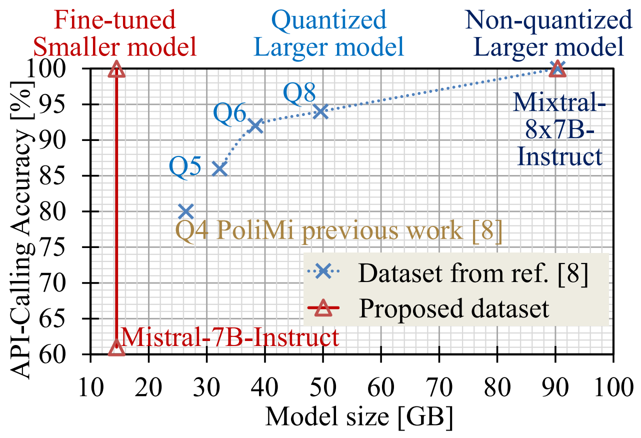
As shown in Fig. 6, we deploy and fine-tune an LLM on the local GPU server and deploy the DT-based SDN controller on the same server to build the AI-Agent A1 for intent-based optical network automation. The SDN controller provides different JSON format APIs written in Python. The LLM is fine-tuned based on API-calling chat prompts and can generate valid predefined APIs in JSON format (compliance to standards is possible but is out of the scope of this paper). Hence, the AI-Agent A1 leverages such an LLM to translate human language requests into valid APIs with correct arguments so that SDN controller can implement them to realize the user's intent.

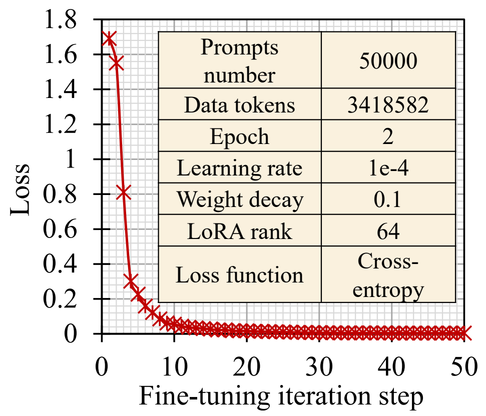
There are five APIs used in controlling , as shown in Tab. 2. The API functions utilize a structured set of arguments to define their inputs and operations. The arguments include:

source and destination: Represent the names of the source and destination nodes in the network. These are required to specify the endpoints for the service and are denoted by string values such as "A" or "B".

path: Specifies the light path for the service. It can be explicitly provided (e.g., "OMS1-OMS2"); if the path is not given explicitly, then it is determined dynamically by a routing and wavelength assignment (RWA) algorithm based on the shortest path of the light path length.

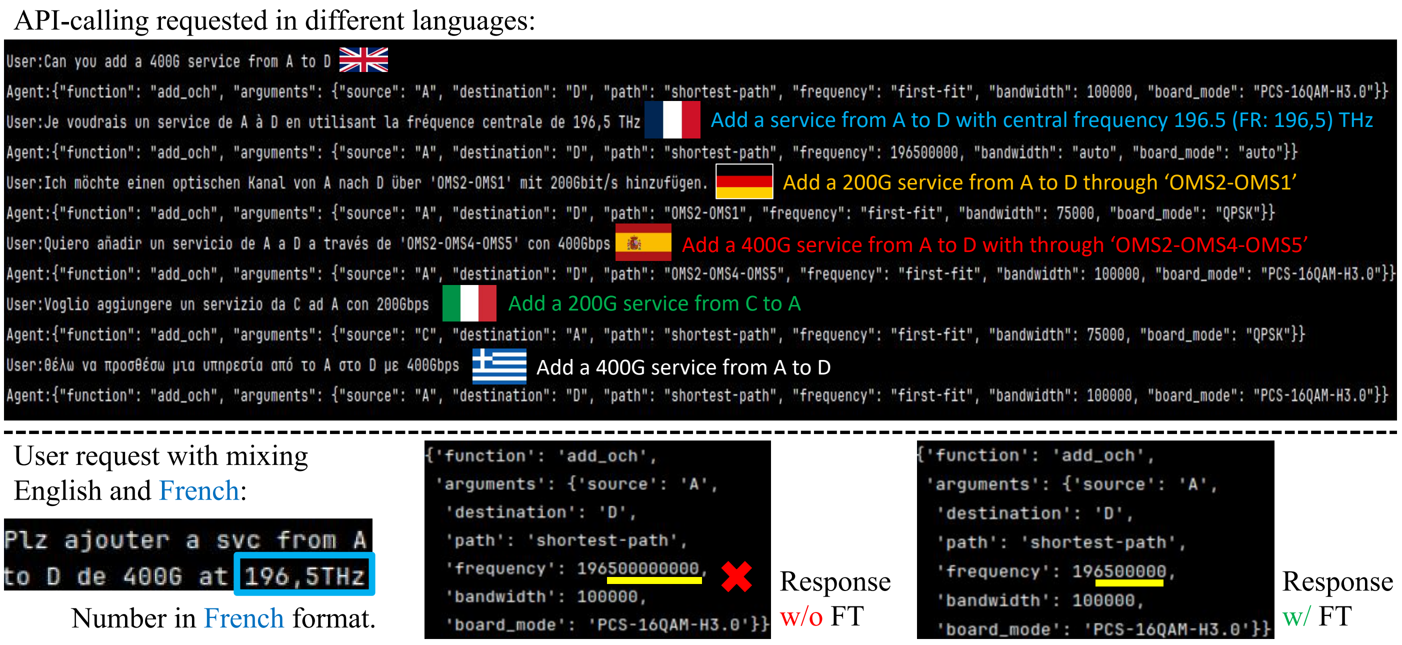
frequency: Indicates the preferred frequency for the service in MHz (e.g., 196625000 [MHz]). If not specified, the RWA al-





Prompts number	50000
Data tokens	3418582
Epoch	2
Learning rate	1e-4
Weight decay	0.1
LoRA rank	64
Loss function	Cross-entropy

Fig. 7. API-calling accuracy with different LLMs (left) and fine-tuning loss curve (right).



```
User:Can you add a 400G service from A to D
Agent:{"function": "add_och", "arguments": {"source": "A", "destination": "D", "path": "shortest-path", "frequency": "first-fit", "bandwidth": 100000, "board_mode": "PCS-16QAM-H3.0"}}
User:Je voudrais un service de A à D en utilisant la fréquence centrale de 196,5 THz
Agent:{"function": "add_och", "arguments": {"source": "A", "destination": "D", "path": "shortest-path", "frequency": 196500000, "bandwidth": "auto", "board_mode": "auto"}}
User:Ich möchte einen optischen Kanal von A nach D über 'OMS2-OMS1' mit 200Gbit/s hinzufügen.
Agent:{"function": "add_och", "arguments": {"source": "A", "destination": "D", "path": "OMS2-OMS1", "frequency": "first-fit", "bandwidth": 75000, "board_mode": "QPSK"}}
User:Quiero añadir un servicio de A a D a través de 'OMS2-OMS4-OMS5' con 400Gbps
Agent:{"function": "add_och", "arguments": {"source": "A", "destination": "D", "path": "OMS2-OMS4-OMS5", "frequency": "first-fit", "bandwidth": 100000, "board_mode": "PCS-16QAM-H3.0"}}
User:Voglio aggiungere un servizio da C ad A con 200Gbps
Agent:{"function": "add_och", "arguments": {"source": "C", "destination": "A", "path": "shortest-path", "frequency": "first-fit", "bandwidth": 75000, "board_mode": "QPSK"}}
User:Θέλω να προσθέσω μια υπηρεσία από το A στο D με 400Gbps
Agent:{"function": "add_och", "arguments": {"source": "A", "destination": "D", "path": "shortest-path", "frequency": "first-fit", "bandwidth": 100000, "board_mode": "PCS-16QAM-H3.0"}}
```

Add a service from A to D with central frequency 196.5 (FR: 196,5) THz
Add a 200G service from A to D through 'OMS2-OMS1'
Add a 400G service from A to D with through 'OMS2-OMS4-OMS5'
Add a 200G service from C to A
Add a 400G service from A to D
User request with mixing English and French:
Number in French format.
Response w/o FT
Response w/ FT

```
Plz ajouter a svc from A
to D de 400G at 196,5THz
```

```
{'function': 'add_och',
 'arguments': {'source': 'A',
  'destination': 'D',
  'path': 'shortest-path',
  'frequency': 196500000000,
  'bandwidth': 100000,
  'board_mode': 'PCS-16QAM-H3.0'}}
```

```
{'function': 'add_och',
 'arguments': {'source': 'A',
  'destination': 'D',
  'path': 'shortest-path',
  'frequency': 196500000,
  'bandwidth': 100000,
  'board_mode': 'PCS-16QAM-H3.0'}}
```

Fig. 8. Multi-language API-calling test. Top: Asking a fine-tuned LLM to generate JSON format API configuration using different languages. Bottom: Asking a confusing question to LLM without (w/o) and with (w/) fine-tuning (FT).

gorithm selects a frequency using a "first-fit" approach, starting from longer wavelengths to shorter wavelengths.

bandwidth: Defines the bandwidth required for the service in MHz, correlating to the board mode. For instance, a 400 Gb/s service requires 100000 [MHz], while a 200 Gb/s service requires 75000 [MHz].

board_mode: Specifies the modulation format of the board. Common options include "PCS-16QAM" for a 400 Gb/s service, "QPSK" for a 200 Gb/s service, or "auto" to dynamically select an available board.

oms_id: Identifies the Optical Multiplex Section (OMS) involved in certain operations, such as quality of transmission (QoT) estimation or power equalization. Examples include "OMS1" or "ALL" to refer to all OMSs.

service_id: Acts as an index to identify a specific service in the network for operations like QoT estimation or measurement.

metric: Refers to the parameter being evaluated or optimized, such as "OSNR", "GSNR", "SNR" (Signal-to-Noise Ratio), "ASENL" (ASE to NL Noise Ratio), or "BER" (Bit Error Rate).

method: Describes the approach for operations like power equalization. Common methods include "ASENL" to optimize the SNR by balancing ASE to nonlinear noise ratio, "OSNR" for equalizing OSNR, and "GSNR" for equalizing GSNR.

AI-Agent A1 should call the correct APIs with as high accuracy as possible. However, as discussed in Section 2, there is a trade-off between performance and general LLM size, which relates to computation resource requirements. For instance, quantization of the parameters is a standard strategy for reducing the model's size and inference costs. Ref [8] achieved 80% accuracy by using Mixtral-8x7B-Instruct with 4-bit quantization (Q4). Based on PoliMi's dataset (limited to 50 queries) made available in the same paper [8], we test the model accuracy with different levels of quantization (Q5-Q8: 5-bit to 8-bit quantization). Fig. 7(left) depicts the obtained model size/accuracy trade-off in blue. The results align with intuition, as larger models tend to perform better in terms of accuracy. While [8] (which leverages formal grammars) and fine-tuning both enable standard-compliant outputs, fine-tuning additionally achieves higher ac-

Table 2. APIs used in experiments and prompts for fine-tuning AI-Agent A1.

API name	Usage	Arguments
add_och	Add a service in the network.	source, destination, path, frequency, bandwidth, board_mode
del_och	Delete a service in the network.	source, destination, frequency
estimate_qot	Estimate QoT of a service/all services.	oms_id, service_id, metric
measure_qot	Measure BER or SNR.	service_id, metric
equalize	Equalize power in OMS or network.	oms_id, method

curacy than [8].

We perform LoRA fine-tuning on Mistral-7B-Instruct for the five APIs we introduced in Section 4 C.1. As shown in Fig. 7(left), larger models achieving 100% accuracy can generate diverse data to fine-tune smaller models. For example, we create a few question-answer pairs and use the larger model to expand them with high diversity. For non-confidential data, public cloud LLMs like ChatGPT can be utilized, while private cloud LLMs are employed to augment sensitive data securely. We generate an augmented dataset of 10k/API for fine-tuning; note that the LLM used to generate the dataset is completely independent of the LLM used in our AI agents, to avoid introducing any bias in the evaluation. The loss curve is shown in Fig. 7(right). We reduce the model size by 83%, from 90.4 GB to 14.5 GB, while maintaining 100% accuracy in API calls, as shown in Fig. 7(left,red).

In addition, another advantage of LoRA fine-tuning is that it only modifies a small portion of the parameters (here, 2.9%), therefore, the model keeps its general abilities, for instance, the multi-language ability. The Mistral AI models support several European languages and still work with requests in different languages although the training set in English. As shown in Fig. 8(top), we use different languages (English, French, German, Spanish, Italian, Greek) to request different services, and the fine-tuned LLM gives us correct API calls in JSON format. Moreover, the fine-tuned model is even robust to confusing questions with mixing languages. For instance, we request the service with indicating frequency in French format, which uses "," instead of ".", yields the non-fine-tuned LLM misunderstands the value/unit (Fig. 8 (bottom)).

C.2. Documentation and Log Analysis

The LLM develops a specialized understanding of commercial optical products content by leveraging either RAG or fine-tuning. As shown in Fig. 9, we leverage these two techniques to build two types of AI-Agent A2 to integrate optical product documentation. To implement RAG, we use LangChain [29] to split the text from documentation bodies into chunks, which are then vectorized and stored in a database with Chroma [30].

By integrating the description of logs found in product manuals, AI-Agent A2 can play a transformative role in analyzing alarm logs from optical networks, a critical aspect of maintaining network reliability and performance. Optical networks generate large alarm logs to indicate potential faults, performance degra-

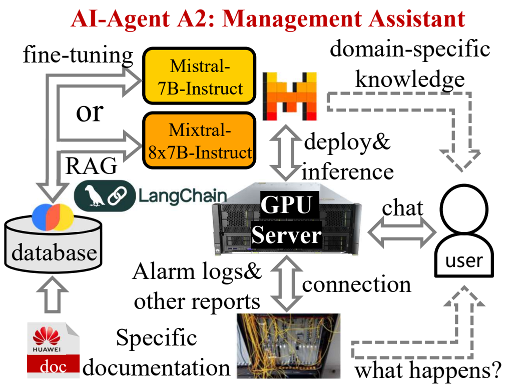


Fig. 9. AI-Agent A2 for network management leverages knowledge from a database fed with product documentation.

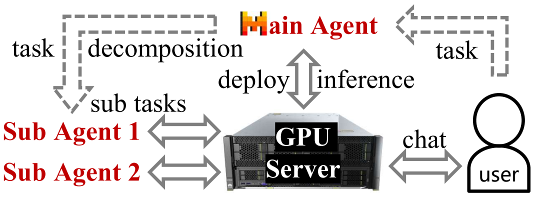


Fig. 10. Multi-agent interaction, on the same GPU server.

dations, or operational anomalies. Reviewing these logs is time-consuming and error-prone, especially in complex, multi-vendor environments. LLMs can automate this process by identifying patterns, classifying alarms, and providing actionable insights.

With the help of AI-Agent A2, engineers can quickly obtain troubleshooting steps, configuration guides, or detailed product specifications tailored to their queries. We will demonstrate RAG for product searching in Section 5 A, alarms management in Section 5 C.2, and fine-tuning for document integration in Section 5 D.2. After showing the results, we compare these two techniques in Section 6.

C.3. Multi-Agent

Since we deploy different AI-Agents on the same GPU server, we let the AI-Agents interact with each other to build a multi-agent system, as shown in Fig. 10. The main agent, which knows the roles of the other agents, is used to communicate with the other agents. An example application is network design (Fig. 11), which will be discussed soon in Section 5.A.

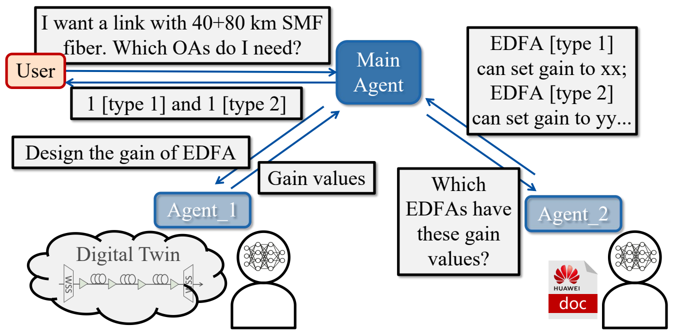

I want a link with 40+80 km SMF fiber. Which OAs do I need?
User
Main Agent
1 [type 1] and 1 [type 2]
EDFA [type 1] can set gain to xx; EDFA [type 2] can set gain to yy...
Design the gain of EDFA
Gain values
Which EDFAs have these gain values?
Agent_1
Agent_2
Digital Twin
doc

Fig. 11. Multi-agent workflow for network design. See Visualization 1 and Visualization 2.

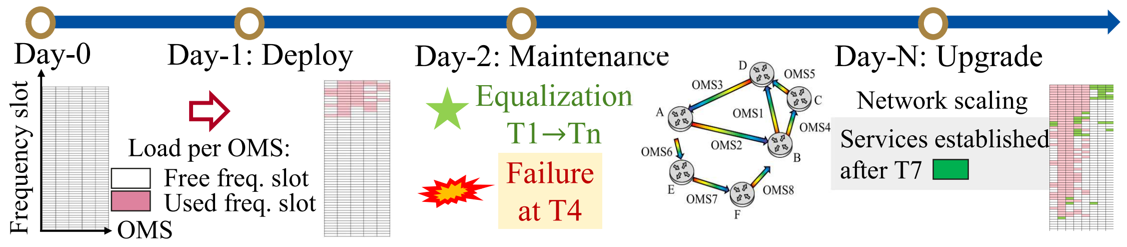


Fig. 12. Lifecycle automation from Day-0 to Day-N.

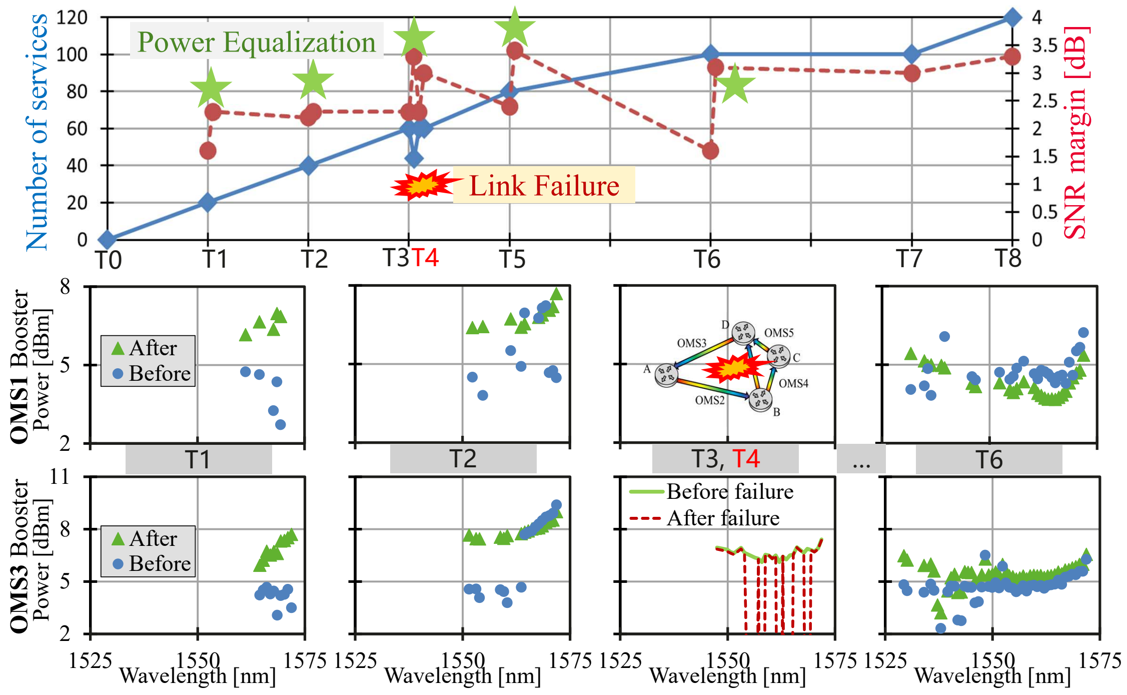


Fig. 13. Operations and power spectra evolution in network lifecycle. Top: number of services (blue) and SNR margin evolution (red). Bottom: booster output spectra for 2 sample OMSes (OMS1 and OMS3) at different lifecycle steps. (Blue circle: before power equalization, green triangle: after power equalization. Green solid line: before link failure, red dash line: after link failure.)

5. LIFECYCLE AUTOMATION

In this section, we demonstrate how AI-Agents can help design/manage an optical network over its lifecycle, from Day-0 to Day-N (see Fig. 12 and Fig. 13).

A. Day-0: Network Design

As shown in Fig. 11, a multi-agent system enables collaboration between specialized agents to streamline network design.

A main AI-Agent is powered by prompt engineering for understanding the roles of different sub-agents. The main agent decomposes the tasks to these sub-agents when the user request requires knowledge or tools from different agents. Besides the APIs to control the optical network, we give AI-Agent A1 here the APIs to use DT for simulating and optimizing physical layer parameters, such as optical amplifier gain configuration. Once the design is complete, the main AI-Agent asks AI-Agent A2 to leverage its knowledge of commercial optical products to recommend suitable components, like the type name of amplifiers, ensuring the design is both technically optimal and implementable with available hardware. We use the JSON format messages for communication between AI-Agents and users, e.g., {"from": "user", "to": "Agent_Main", "content": "xxx"}, {"from": "Agent_Main", "to": "Agent1", "content": "xxx"}. (See Visualization 2.)

This collaborative framework bridges theoretical simulation and practical application, enhancing efficiency and feasibility in optical network development.

B. Day-1: Network Deployment

Once the optical network devices have been deployed, commissioning is required to establish the services. We start from a 5-OMS meshed network and upgrade it to 8-OMS in the end. We use AI-Agent to establish the services automatically. The workflow is shown in Fig. 14 and demonstrated in Visualization 3 :

Step-1: The user requests the services in natural language, for instance, “Please build a 400G service from node A to D”.

Step-2: AI-Agent A1 understands the intent and generates the valid APIs sending to the DT for specific tasks introduced in section 4 C.1.

Step-3 : Then the DT leverages an RWA algorithm to assign the light path routing and frequency slot allocation.

Step-4: AI-Agent A1 gives suggestions to the user and requests approval.

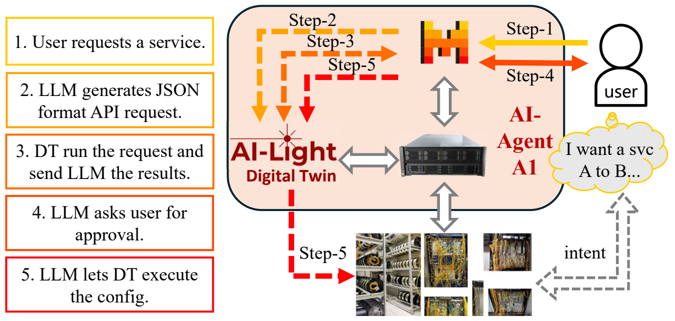

1. User requests a service.
2. LLM generates JSON format API request.
3. DT run the request and send LLM the results.
4. LLM asks user for approval.
5. LLM lets DT execute the config.
Step-1
Step-2
Step-3
Step-4
Step-5
Step-5
AI-Light Digital Twin
AI-Agent A1
user
I want a svc A to B...
intent

Fig. 14. Service establishment workflow. See Visualization 3.

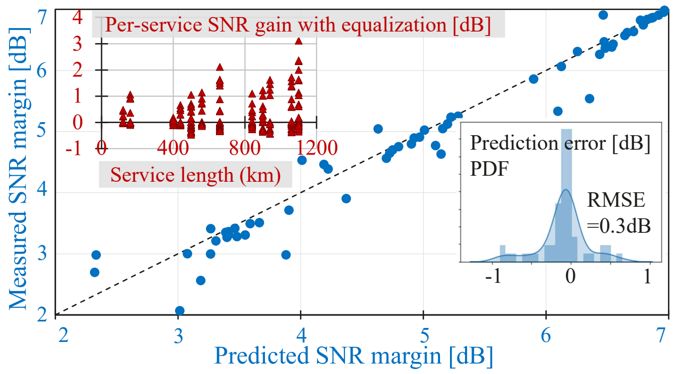


Fig. 15. Digital twin validation: SNR improvement with power equalization and SNR prediction accuracy.

Step-5: Upon approval, AI-Agent A1 uses the DT to configure the transmitter, receiver, and WSSs to establish the service.

C. Day-2: Network Maintenance

For network maintenance, we consider two tasks: power equalization and troubleshooting. To emulate these tasks, we load new batches of 20 services each (see slot occupation in Fig. 13 at T1, T2, T3, T5, T6); as introduced in Section 4 A, the services are emulated by an ASE source and we use a single transponder to replace the ASE channels during the SNR measurements ; for the following numerical evaluations, we measured SNR for 20% of the services.

C.1. Periodic Power Equalization

During the network lifecycle, amplifiers' gains vary with load, such that launch power profiles vary as well. To avoid the resulting SNR margin degradation, we use periodic service launch power re-equalization as in [31].

By chatting with AI-Agent A1, we initially add services according to the "set and forget" method, i.e., each service is established with a fixed channel power, which is not re-optimized as new services are added.

Then, we periodically re-equalize the power through API-calling by AI-Agent A1 to optimize the SNR (up to 1dB margin improvement is measured at T6) by balancing the ASE-to-NL noise ratio. For example, the power spectra before and after equalization at different times are shown in Fig. 13. The measured network SNR margin (defined as the minimum across all services, difference between the service SNR and its FEC limit) before and after re-equalization for each of the five batches is shown in Fig. 13 (jumps in the red dashed line), and the DT predicted SNR gains for all services over network lifetime in

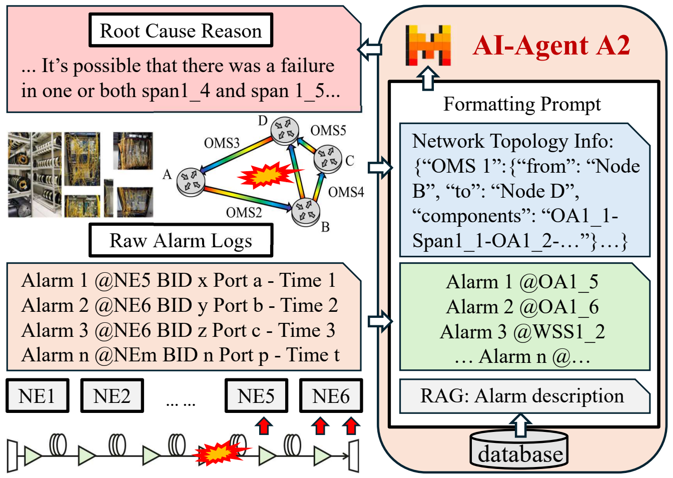

Root Cause Reason
... It's possible that there was a failure in one or both span1_4 and span 1_5...
AI-Agent A2
Formatting Prompt
Network Topology Info: {"OMS 1":{"from": "Node B", "to": "Node D", "components": "OA1_1-Span1_1-OA1_2-..."}...}
Raw Alarm Logs
Alarm 1 @NE5 BID x Port a - Time 1
Alarm 2 @NE6 BID y Port b - Time 2
Alarm 3 @NE6 BID z Port c - Time 3
Alarm n @NEm BID n Port p - Time t
Alarm 1 @OA1_5
Alarm 2 @OA1_6
Alarm 3 @WSS1_2
... Alarm n @...
RAG: Alarm description
database
NE1
NE2
NE5
NE6
OMS2
OMS3
OMS4
OMS5
A
B
C
D

Fig. 16. Log analysis workflow for a link failure (NE: network element; BID: board identifier; OA: optical amplifier).

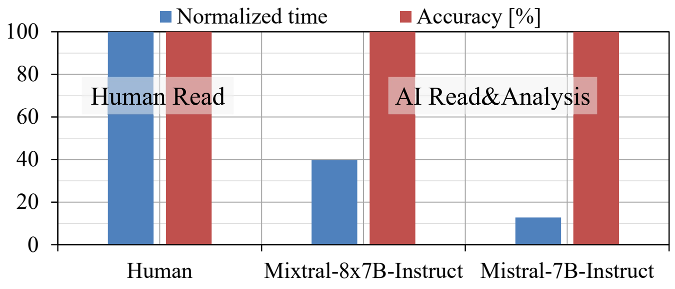


Fig. 17. Log analysis time and accuracy comparison. Time is normalized to human reading time (as 100).

Fig. 15(inset). Note that some services may degrade but the network margin always improves.

The DT performs prediction and estimation of SNR before and after re-equalization. Prediction here is the prediction of the optimized state, based on the monitoring available for the previous non-optimized network state. The probability density function (PDF) of the SNR prediction accuracy ($SNR_{prediction} - SNR_{measured}$) is shown in Fig. 15, the root mean square error (RMSE) is 0.3dB.

C.2. Troubleshooting

Event logs are continuously generated in optical networks, and alarms occur over time. As a natural language processing technique, LLMs are highly effective in alarm-log analysis. Based on the topology of a network and the logs from the network elements, an LLM can figure out where a failure has occurred.

The logs from the network elements contain the time stamp of each event, however, time synchronization across network elements cannot be guaranteed. As a consequence, the event time from different network elements may not be trusted for troubleshooting, and the spatial information, i.e., topology, is more important for troubleshooting. Hence, we also provide the topology information in a formatted prompt. Additionally, raw alarm logs from various network elements capture event details for individual boards, identified by their board ID and port numbers. To enhance analysis, the board ID and port are mapped to the corresponding indices of OMSs and optical amplifiers in the prompt, ensuring precise alignment with the network topology and providing clear contextual relevance. The LLM-enabled

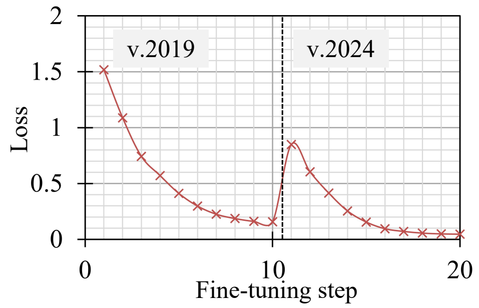


Fig. 18. Commercial products documentation-based fine-tuning before/after upgrade. Version 2019 is for C-band commercial equipment only, while version 2024 updates L-band commercial equipment.

troubleshooting workflow is shown in Fig. 16.

We unplugged a connector to emulate a fiber cut on OMS1 (leading to the drop of 16 services) at T4 and collected the logs, which were then analyzed by AI-Agent A2. As shown in Fig. 17, by comparing with the reading speed of human beings [32], AI-Agent A2 performed alarm log reading and analysis 7x faster than a human would parse the logs (let alone analyze them).

In addition, we also emulate link failure by disabling in turn the amplifier located before each of the 19 spans. The LLM-based AI-Agent A2 troubleshooter successfully returns the location for each of the 19 emulated failures, as shown in Fig. 17. However, we need to observe that, despite different models achieving the same accuracy, we have to tune the prompt for each model. In other words, the same prompt yields different performances depending on the underlying LLM. Moreover, the order of information (topology, alarms, descriptions) also impacts the output of different LLMs.

D. Day-N: Network Upgrade

D.1. Scaling the Network

To avoid blocking at end-of-life (T7), we add 3 OMSs and use AI-Agent A1 to automate the addition of 20 services in the 8-OMS network, using the same method as in Day-2, reaching a total of 120 services established, see Fig. 13 (right).

D.2. Hardware and Software Upgrade

While RAG retrieves relevant information from vast technical repositories, fine-tuning adapts an LLM to deeply understand the structure, terminology, and context of domain-specific documentation. By fine-tuning the model on commercial product manuals, the LLM develops a specialized understanding of commercial optical networks content. This ensures that even without information retrieval, the model can independently provide accurate and context-aware insights.

However, different products and features are typically released during the 10-20 years lifetime of optical equipment. The AI-Agent A2 also requires to be updated. Since the safetensor is just a checkpoint of the training (cf. Section 2), model fine-tuning can be done iteratively on the previous model each time new product documentation or a new software version is released. Safetensors enable continuous fine-tuning using prior checkpoints.

For example, a model initially fine-tuned with documentation from 2019 was subsequently enhanced using updated 2024 documentation, resulting in improved performance while preserving earlier optimizations. The training dataset comprised specifications from various commercial products, including WSSs, EDFAs, transponders, etc. Notably, the 2019 dataset focuses exclusively on C-band equipment, whereas the 2024 dataset incorporates updated specifications for L-band equipment, reflecting technology advancements. The training loss curve (Fig. 18) demonstrates smooth convergence each time fine-tuning is triggered, showcasing the practicality of incremental model upgrades.

Table 3. Comparison between RAG and Fine-tuning.

Technique	RAG	Fine-tuning
Pros	Dynamic knowledge access, reduced training costs, scalability.	Fast response on specific, static knowledge base; make smaller models powerful and consistent.
Cons	Slower inference; complexity in integration.	Re-training is needed each time the knowledge changes.

Table 4. Comparison of LLMs, Traditional machine learning (ML), and Rule-Based Systems in Optical Network Management and Control.

Technique	LLMs	Traditional ML	Rule-Based Systems
Flexibility	High, adaptable to various tasks	Moderate	Low, hard-coded logic
Interpret-ability	Low, black-box models	Moderate, statistical models	High, explicit rules
Scalability	High, but resource-intensive	Moderate	Low, manual scaling required
Ease of use	High, natural language interface	Moderate	Low, technical expertise required

6. DISCUSSION

In this Section, we discuss a series of open issues that arose during the implementation of this demonstration and that point at important future research directions in this area.

RAG vs. Fine-Tuning. In Section 5, we demonstrate that AI-Agent 2 can work with either RAG or fine-tuning. Different tasks in optical network control and management can be done by either RAG or fine-tuning. The choice between RAG and fine-tuning depends on the specific requirements of the task. As listed in Tab. 3, RAG is suitable for applications that require dynamic and up-to-date information, while fine-tuning is better for tasks that need consistent and coherent responses within a specific domain. Each approach has its trade-offs, and the decision should be based on the available resources, the nature of the task, and the desired outcome.

Table 5. Pros and Cons of LLMs on Control (e.g., service deployment, optimization) and Management (e.g., troubleshooting, knowledge search).

Application	Control	Management
Pros	Intent-based networking. Reduces manual intervention, minimizing human error. Speeds up the network configurations.	Quickly analyzes large amounts of network logs. Reduces the need for highly specialized knowledge among network operators.
Cons	Requires accurate DT. Requires high computational resources. Adds latency for time-sensitive app.	Data privacy issue. Hallucinations: false positives or negatives. Requires continuous training.

LLM vs. Exisiting Solutions. While LLMs offer promising capabilities for automating optical network control and management, a comprehensive comparison among LLMs, traditional machine learning, and rule-based systems is yet to be thoroughly conducted. Tab. 4 provides a qualitative comparison of LLMs, traditional machine learning (ML), and rule-based systems in terms of flexibility, interpretability, scalability, and ease of use. It highlights that LLMs offer high flexibility and a natural language interface but are resource-intensive and less interpretable compared to rule-based systems, which are rigid but highly interpretable.

Tab. 5 outlines the pros and cons of LLM-based techniques specifically for control and management applications. For control, LLMs enable intent-based networking and reduce manual intervention but require accurate data and significant computational resources. In management, LLMs excel at analyzing network logs and reducing the need for specialized knowledge among operators, though they raise concerns about data privacy and continuous training requirements.

Although LLMs present a powerful tool for enhancing network automation throughout the lifecycle, it is essential to consider the critical requirements related to data accuracy, computational resources, and interpretability during their deployment.

Data Privacy. As data privacy is a critical requirement for operators, we decided to deploy our LLM locally instead of on the public cloud in this work. However, this approach involves additional expenses and expertise for fine-tuning the model and maintaining local servers and the related infrastructure.

Computational Requirements. Training, fine-tuning, and even RAG require large amounts of high-quality data, and LLMs are computation-hungry, so there are concerns about the costs and return on investment of training the LLMs. While the topic of computational cost is still debated, we notice that new techniques to improve the efficiency of training the LLMs are emerging, e.g., distillation (recently used by DeepSeek-R1 [33]).

Reliability. Optical networking is a field that requires high reliability, therefore, interpretability is highly desired. However, LLMs are black-box models, and prompt engineering does not permit consistent output, hindering explainability and reducing trust in LLM-based network operation. For this reason, we propose to leverage DT with accurate physical tools and interpretable workflow for the physical-layer related operations, thereby achieving a predictive control loop, i.e., perception-prediction-action-feedback.

Further research is needed to fully understand the strengths and limitations and quantify the performance of LLMs in this context relative to existing automation techniques.

7. CONCLUSIONS AND FUTURE PERSPECTIVES

This study demonstrates a successful application of locally fine-tuned LLMs and DT technologies for the comprehensive lifecycle management of optical networks. Our framework achieves efficient automation by addressing key tasks for network design, deployment, maintenance, and upgrade while ensuring data privacy through on-premises deployment. Experimental results illustrate substantial gains in intent interpretation accuracy, and efficiency, underlining the feasibility of this approach in real-world scenarios. Moreover, integrating multi-agent systems and advanced AI techniques ensures adaptability and scalability to evolving network demands. Future research will focus on enhancing model robustness, expanding domain adaptability, exploring advanced functionalities, and improving efficiency on various tasks, paving the way for autonomous optical networks.

REFERENCES

1. Y. Wang, C. Zhang, J. Li, Y. Pang, L. Zhang, M. Zhang, and D. Wang, “AlarmGPT: an intelligent alarm analyzer for optical networks using a generative pre-trained transformer,” J. Opt. Commun. Netw. **16**, 681–694 (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,” Electronics. **13** (2024).
3. X. Jiang, M. Zhang, Y. Song, Y. Zhang, Y. Wang, C. Ju, and D. Wang, “OptiComm-GPT: a GPT-based versatile research assistant for optical fiber communication systems,” Opt. Express **32**, 20776–20796 (2024).
4. Y. Pang, M. Zhang, Y. Liu, X. Li, Y. Wang, Y. Huan, Z. Liu, J. Li, and D. Wang, “Large language model-based optical network log analysis using LLaMA2 with instruction tuning,” J. Opt. Commun. Netw. **16**, 1116–1132 (2024).
5. D. Adanza, C. Natalino, L. Gifre, R. Muñoz, P. Alemany, P. Monti, and R. Vilalta, “IntentLLM: An AI Chatbot to Create, Find, and Explain Slice Intents in TeraFlowSDN,” in *2024 IEEE 10th International Conference on Network Softwarization (NetSoft),* (2024), pp. 307–309.
6. A. Zhou, Y. Song, Y. Zhang, M. Zhang, and D. Wang, “Large Language Model-Driven AI Agent in SDN Controller Towards Intent-Based Management of Optical Networks,” in *50th European Conference on Optical Communication (ECOC) 2024,* (2024).
7. Y. Song, Y. Zhang, A. Zhou, Y. Shi, S. Shen, X. Tang, J. Li, M. Zhang, and D. Wang, “Synergistic Interplay of Large Language Model and Digital Twin for Autonomous Optical Networks: Field Demonstrations,” IEEE Commun. Mag. (2024).
8. N. D. Cicco, M. Ibrahimi, S. Troia, F. Musumeci, and M. Tornatore, “Open Implementation of a Large Language Model Pipeline for Automated Configuration of Software-Defined Optical Networks,” in *50th European Conference on Optical Communication (ECOC) 2024,* (2024).
9. V. Karunakaran, C. Natalino, B. Shariati, P. Lechowicz, J. K. Fischer, A. Autenrieth, P. Monti, and T. Bauschert, “TAPI-based Telemetry Streaming in Multi-domain Optical Transport Network,” in *Optical Fiber Communication Conference (OFC) 2024,* (Optica Publishing Group, 2024), p. M3Z.9.
10. E. Etezadi, C. Natalino, V. Karunakaran, R. Diaz, A. Lindgren, S. Melin, A. Autenrieth, L. Wosinska, P. Monti, and M. Furdek, “Demonstration of DRL-based intelligent spectrum management over a T-API-enabled optical network digital twin,” in *49th European Conference on Optical Communications (ECOC 2023),* vol. 2023 (2023), pp. 1730–1733.

11. D. Wang, Y. Song, Y. Zhang, X. Jiang, J. Dong, F. N. Khan, T. Sasai, S. Huang, A. P. T. Lau, M. Tornatore, and M. Zhang, “Digital Twin of Optical Networks: A Review of Recent Advances and Future Trends,” J. Light. Technol. **42**, 4233–4259 (2024).
12. Q. Zhuge, X. Liu, Y. Zhang, M. Cai, Y. Liu, Q. Qiu, X. Zhong, J. Wu, R. Gao, L. Yi, and W. Hu, “Building a digital twin for intelligent optical networks [Invited Tutorial],” J. Opt. Commun. Netw. **15**, C242–C262 (2023).
13. R. Vilalta, L. Gifre, R. Casellas, R. Muñoz, R. Martínez, A. Mozo, A. Pastor, D. López, and J. P. Fernández-Palacios, “Applying Digital Twins to Optical Networks with Cloud-Native SDN Controllers,” IEEE Commun. Mag. **61**, 128–134 (2023).
14. M. S. Faruk and S. J. Savory, “Measurement Informed Models and Digital Twins for Optical Fiber Communication Systems,” J. Light. Technol. **42**, 1016–1030 (2024).
15. Y. He, Z. Zhai, L. Dou, L. Wang, Y. Yan, C. Xie, C. Lu, and A. P. T. Lau, “Improved QoT estimations through refined signal power measurements and data-driven parameter optimizations in a disaggregated and partially loaded live production network,” J. Opt. Commun. Netw. **15**, 638–648 (2023).
16. M. Devigili, M. Ruiz, N. Costa, C. Castro, A. Napoli, J. Pedro, and L. Velasco, “Applications of the ocata time domain digital twin: from qot estimation to failure management,” J. Opt. Commun. Netw. **16**, 221–232 (2024).
17. C. Sun, X. Yang, N. D. Cicco, R. Ayassi, V. V. Garbhapu, P. A. Stavrou, M. Tornatore, G. Charlet, and Y. Pointurier, “First Experimental Demonstration of Full Lifecycle Automation of Optical Network through Fine-Tuned LLM and Digital Twin,” in *50th European Conference on Optical Communication (ECOC) 2024,* (2024), p. Th3B.6.
18. A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention Is All You Need,” (2017).
19. OpenAI, “ChatGPT: Large Language Model,” https://openai.com (2024).
20. H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, J.-P. Guillou, S. Schmidt, and M. Auli, “Llama: Open and efficient foundation language models,” (2023).
21. Mistral AI, https://mistral.ai (2024).
22. E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, “LoRA: Low-Rank Adaptation of Large Language Models,” (2021).
23. K. Christodoulopoulos, C. Delezoide, N. Sambo, A. Kretsis, I. Sartzetakis, A. Sgambelluri, N. Argyris, G. Kanakis, P. Giardina, G. Bernini, D. Roccato, A. Percelsi, R. Morro, H. Avramopoulos, P. Castoldi, P. Layec, and S. Bigo, “Toward efficient, reliable, and autonomous optical networks: the ORCHESTRA solution [Invited],” J. Opt. Commun. Netw. **11**, C10–C24 (2019).
24. P. Poggiolini, G. Bosco, A. Carena, V. Curri, Y. Jiang, and F. Forghieri, “The GN-Model of Fiber Non-Linear Propagation and its Applications,” J. Light. Technol. **32**, 694–721 (2014).
25. A. Ferrari, V. V. Garbhapu, D. L. Gac, I. F. de Jauregui Ruiz, G. Charlet, and Y. Pointurier, “Demonstration of AI-Light: an Automation Framework to Optimize the Channel Powers Leveraging a Digital Twin,” in *Optical Fiber Communication Conference (OFC) 2022,* (Optica Publishing Group, 2022), p. M3Z.14.
26. Mistral AI, “Mistral-7b-instruct-v0.3,” https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.3.
27. Mistral AI, “Mixtral-8x7b-instruct-v0.1,” https://huggingface.co/mistralai/Mistral-8x7B-Instruct-v0.1.
28. Hugging Face, “Hugging Face,” https://huggingface.co.
29. LangChain, “Langchain,” https://github.com/langchain-ai/langchain.
30. Chroma, “Chroma,” https://github.com/chroma-core/chroma.
31. X. Yang, A. Ferrari, D. L. Gac, G. Charlet, M. Tornatore, and Y. Pointurier, “Experimental impact of power re-optimization in a mesh network,” J. Opt. Commun. Netw. **15**, C20–C28 (2023).
32. M. Brysbaert, “How many words do we read per minute? A review and meta-analysis of reading rate,” J. Mem. Lang. **109**, 104047 (2019).
33. DeepSeek-AI, “DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning,” (2025).