International Journal of Network Management

WILEY

RESEARCH ARTICLE OPEN ACCESS

A Comprehensive Survey on LLM-Based Network Management and Operations

Jibum Hong | Nguyen Van Tu | James Won-Ki Hong

Department of Computer Science and Engineering, POSTECH, Pohang, Korea

Correspondence: Jibum Hong (hosewq@postech.ac.kr)

Received: 1 April 2025 | **Revised:** 18 September 2025 | **Accepted:** 24 September 2025

Funding: This work was supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (RS-2024-00392332, Development of 6G Network Integrated Intelligence Plane Technologies).

ABSTRACT

The growing demands for network capacity and the increasing complexities of modern network environments pose significant challenges for effective network management and operations. In response, network operators and administrators are moving beyond traditional manual and rule-based methods, adopting advanced artificial intelligence (AI)-driven paradigms (e.g., self-driving networks, autonomous networks, network automation). Recently, large language models (LLMs) have emerged as a promising AI technology with the potential to revolutionize network management and operations through natural language interaction. In this paper, we provide a comprehensive survey of LLM-based approaches in network management and compare those approaches with existing methods. We identify key advantages of LLM-based approaches, such as their ability to interpret intent and automate complex tasks, as well as limitations, which include hallucinations and domain adaptation challenges. Based on these insights, we outline open technical challenges and propose future research directions to guide the development of LLM-based network management.

1 | Introduction

The scale and complexity of networks have grown significantly in recent years. Modern network infrastructures must support increased traffic demand and heterogeneous environments, which have led to many challenges in network management [1]. While techniques such as virtualization (e.g., deploying network functions as virtual machines (VMs) or containers through network function virtualization (NFV) [2] and software-defined networking (SDN) [3]) improve flexibility and reduce hardware costs, they also introduce additional complexity [4]. The mix of physical and virtual network components results in diverse configurations that are difficult for administrators to fully understand and manage.

Traditional network management approaches, which use primarily manual operations or rigid rule-based policies executed by network administrators, have become inadequate in this context. These legacy methods lack the agility to adapt to rapidly changing network conditions or scale with the growing size of networks. As a result, network operators face increasing difficulty in maintaining performance and reliability using conventional tools.

To address these issues, network operators and service providers have turned to artificial intelligence (AI) techniques for network management [5, 6]. Advances in AI, especially in machine learning (ML), deep learning (DL), and reinforcement learning (RL), have enabled a new generation of network management solutions. Numerous studies have proposed AI-driven methods to automate network lifecycle management, including intelligent configuration, resource allocation, scaling, and migration [7, 8]. With these methods, progress in natural language processing (NLP) has led to intent-based networking (IBN) [9], which allows network operators to specify high-level requirements in natural language. IBN systems translate these abstracted intents into network policies, empowering even non-expert administrators to configure and operate networks with minimal specialized knowledge [9, 10].

Most recently, large language models (LLMs) [11] have attracted tremendous attention for their potential to further advance network management and orchestration. LLMs offer a distinctive capability compared with earlier ML/DL approaches: they can learn from unstructured text data (such as documentation, configuration files, and logs) and understand natural language instructions [12]. An LLM can interpret a request from a network administrator expressed in plain natural language and automatically generate the necessary configuration or policy to fulfill the request. For example, if an operator says, “Block all YouTube traffic to the office,” an LLM can parse this intent and suggest appropriate changes to access control lists (ACLs) or quality of service (QoS) parameters. When combined with paradigms such as chat-based operations (ChatOps) or IBN, LLM-based tools enable even nonexperts to control complex network environments through natural language, reducing human errors and improving operational efficiency.

Beyond network configuration, LLMs are proving useful in network monitoring and troubleshooting. LLMs can summarize large volumes of log and event data, analyze meaningful cause-and-effect relationships, and assist administrators in diagnosing issues. Whereas rule-based systems often fail to detect anomalies outside of predefined patterns, an LLM's context understanding enables it to identify novel issues and infer root causes. For example, given a series of unusual log messages, LLMs recognize a pattern indicative of a specific network attack or congestion, and then recommend possible mitigation steps (such as security policy updates or traffic rerouting).

Despite their promise, applying LLMs to network management also introduces new challenges. First, LLMs are known to sometimes generate incorrect or fabricated outputs (i.e., “hallucination” problem), which in a network context could lead to faulty configurations if not checked [13]. Second, LLMs are resource-intensive, often requiring a substantial amount of computing power and memory, which raises concerns about deployment in real-time network environments. Third, there is a scarcity of network-specific training data for LLMs, making it difficult to fine-tune LLMs for networking tasks. Fourth, security and privacy issues arise as well: using sensitive configuration files or operational logs to train or prompt an LLM could expose critical information if not handled properly. However, the potential benefits of LLM-driven automation, such as significantly elevating the level of network orchestration and enabling natural language-based operations, make this a compelling research direction.

In this paper, we discuss the state-of-the-art in LLM-based network management and operations. We examine recent studies that use LLMs for various network management tasks compared with existing studies. We divide network management tasks into network design, configuration, fault management, security, and orchestration (including automation and optimization) [7, 8]. Through this, we highlight the advantages of LLM-based approaches, in which they outperform ML/DL or rule-based approaches, as well as the current limitations of LLM-based approaches. We also discuss the technical challenges that must be addressed to effectively deploy LLMs in real-world networks. Finally, we present future research directions to advance LLM-based network management.

Table 1 illustrates the comparisons between our work and existing LLM-based surveys. Existing surveys focus on LLM-based networking, network management, or optimization with different network environments (telecommunications, mobile, cloud, IoT, vehicular, etc.). These surveys usually propose general explanations for LLM and theoretical concepts for the adaptation of the network domain [14, 15, 19]. However, some surveys [14, 15] provide insufficient case studies for the network domain. These surveys introduce related work in other fields that are applicable to network management rather than studies that use actual LLM-based approaches. Furthermore, the other surveys [19, 20] mainly cover the early language models based on Transformer [12] or BERT [21].

Our work mainly surveys LLM-based use cases comprehensively and introduces the latest studies that are not included in existing surveys. We analyze the methods, contributions, limitations, and differences from existing approaches proposed by each study through case-by-case analysis. Finally, we discuss the adaptation of LLM in network management.

The summary of our contributions is as follows:

- **Comprehensive overview of LLM concepts for networking**: We introduce the concepts of LLMs and provide background on how LLMs can be applied to network management and orchestration. This includes bridging the gap between networking requirements and the capabilities of LLM.
- **Survey and analysis of LLM-based network management approaches**: We review existing studies and implementations which use LLM in network management. We analyze the advantages of these approaches offer (e.g., easier configuration and intelligent troubleshooting), as well as their limitations (e.g., hallucinations and real-time processing constraints). This analysis provides a foundation for understanding the current state of LLM-based network management and identifying areas for improvement.
- **Discussion of challenges and future directions**: We discuss the technical challenges in adopting LLMs for network management (e.g., needs for domain-specific LLM training and reliable designs). We also explore how emerging technologies can be combined with LLM. Based on these insights, we outline a roadmap for evolving the network management ecosystem through LLM adaptation.

Figure 1 describes the composition of this survey. The remainder of the manuscript is organized as follows. First, we review the background and related work on LLMs and AI-based network management in Section 2. Second, we leverage the applicable tasks by LLMs for network management in Section 3. Third, we analyze and compare the existing studies and solutions to applying LLMs for network management tasks in Section 4.

TABLE 1 | Comparison of existing LLM-based network management surveys.

References	LLM fundamentals	Domain adaptation methodology	Target domain	Main focus, scope, and taxonomy
Mayer [14] (2024)	X	O	Cloud networks	AI and LLM adaptation for cloud networking. Provide conceptual discussions on LLM adaptation.
C. Liu et al. [15] (2025)	X	O	Communication networks	LLM adaptation for networking. Applications for network design, configuration, fault management, and security. Provide conceptual methodology for adaptation with few case studies.
Qu et al. [16] (2024)	O	X	Mobile edge networks	LLM adaptation for mobile edges. Applications for edge training, caching, inference, and delivery.
Xu et al. [17] (2024)	X	X	IoT networks	Integration of Mixture of Experts and Generative AI in IoT. Applications for transmission scheduling, network slicing, anomaly detection, and autonomous driving.
Javaid et al. [18] (2024)	O	X	UAV networks	LLM adaptation for UAVs. Applications for disaster management, delivery service, logistics, and monitoring.
Long et al. [19] (2025)	O	O	Communication networks	LLM adaptation for Network operation and optimization (configuration, fault management, security, optimization). Mostly composed of ML/DL, RL and BERT-based studies.
Zhou et al. [20] (2025)	O	X	Communication networks	LLM applications for generation, classification, optimization, and prediction. Mostly composed of ML/DL, RL and BERT-based studies.
Our work	O	O	Communication networks	LLM adaptation for network management. Applications for network design, configuration, fault management, security, orchestration.

Fourth, we discuss limitations, technical challenges, and future research directions in Section 5. Finally, we summarize and conclude this paper in Section 6. The abbreviations are summarized in the appendix.

2 | Background

In this section, we present the background on AI-driven network management and the emergence of LLMs in the network domain. We first review traditional network management practices and recent AI-based enhancements, including ML/DL, RL, and IBN. We then introduce the fundamentals of LLMs and discuss techniques for adapting them to network management tasks.

2.1 | AI-Based Network Management

2.1.1 | Overview

Traditional network management has been mainly dependent on standardized, rule-based policies and human expertise [8]. Commonly used approaches have included the use of protocols such as SNMP [22] for device monitoring, tools such as NetFlow [23] for traffic analysis, and manual or script-driven

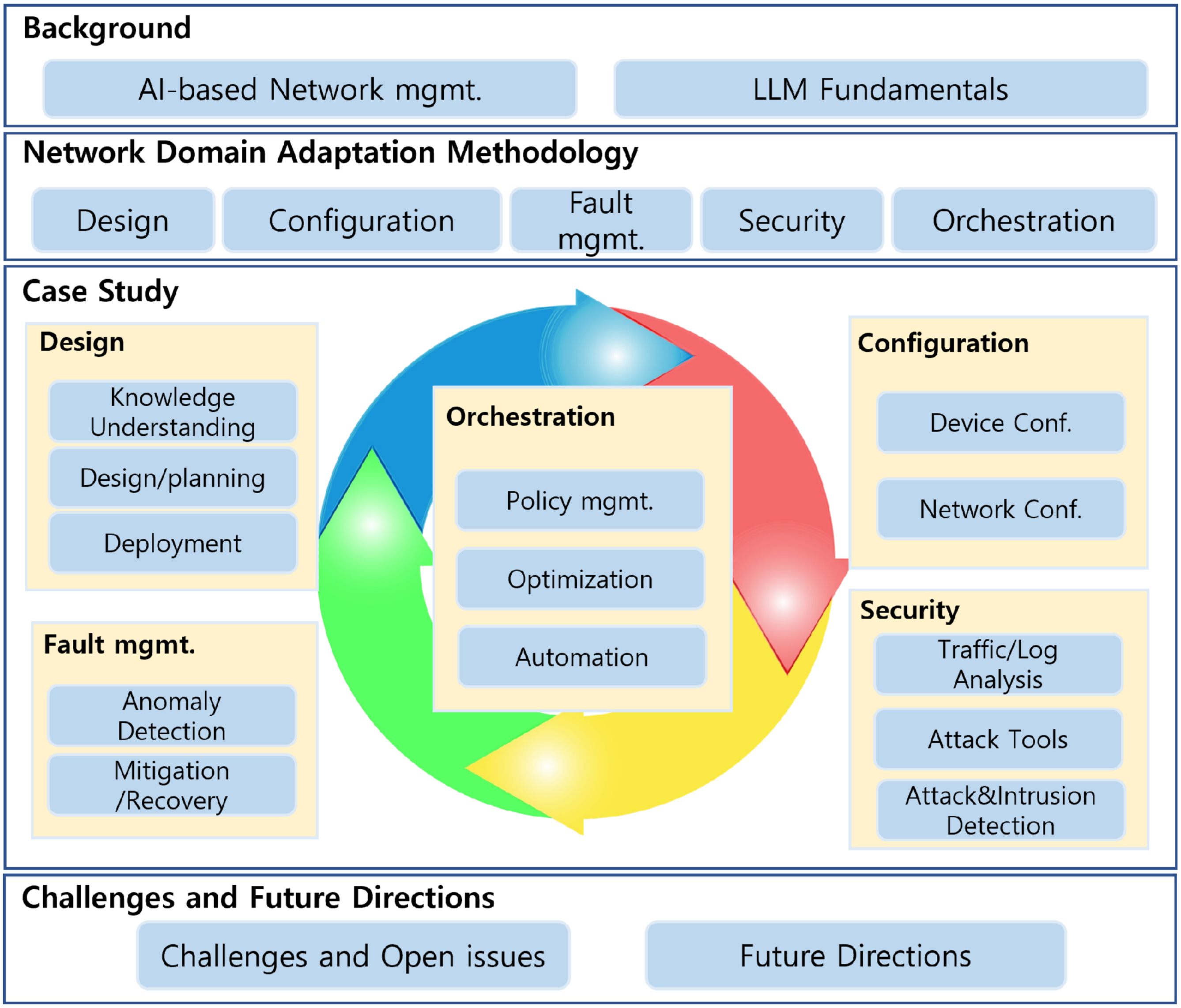


FIGURE 1 | Composition of the proposed survey.

processes for fault diagnosis and restoration. These conventional approaches, while well established, suffer from several limitations [7]:

- **Fixed policies and rules**: Traditional systems operate on predefined static rules. The systems struggle with conditions which are outside of their predefined policies, lacking flexibility when the network environment changes from expected patterns.
- **Passive management**: Traditional management methods are often reactive. When a fault or anomaly occurs, restoration or mitigation methods are typically based on manual intervention by network administrators, which makes real-time responses difficult.
- **Limited scalability**: As networks grow in size and complexity, manual and rule-based management becomes increasingly untenable. The operational costs and risk of errors increase with the scale and diversity of the network.

These limitations have prompted a strong incentive to introduce new technologies that provide greater autonomy, real-time responsiveness, and predictive management capabilities in physical and virtual networks [2, 24].

Over the past decade, AI techniques have been increasingly adopted in network management and orchestration to meet these needs. Early efforts involved knowledge-based systems (e.g., expert systems), but these methods have become insufficient in dynamic network environments and could not handle the large volume of data produced by complex and large networks. Consequently, the industry shifted toward data-centric learning approaches, taking advantage of the ability of AI algorithms to train from network datasets. AI-driven approaches are transforming network management in several ways [7, 25]:

- **Data-driven decision making**: Machine learning models can analyze vast amounts of network data (logs, telemetry, traffic statistics, etc.) in real time, enabling capabilities such as anomaly detection, traffic prediction, and proactive resource optimization, which surpass static rule sets.

- **Automation and autonomous operation**: AI models can automatically perform tasks such as fault diagnosis and recovery, configuration updates, and performance optimization. These reduce the needs for constant human oversight and minimizes manual errors, leading to more efficient and reliable operations.
- **Integration with advanced networks**: AI techniques are increasingly integrated with advanced network environments or technologies such as SDN/NFV, MEC (Multi-access Edge Computing), and 5G/B5G [26] to enhance network management [8, 27]. In 5G/B5G networks, AI-driven optimization dynamically adjusts network slices and virtualized network functions (VNFs) to ensure efficient routing and QoS. Similarly, in MEC [28] environments, AI-based approaches optimize edge resource allocation for latency-sensitive applications such as autonomous driving and augmented reality (AR) [29]. This integration enables adaptive and intelligent network management.

Through these capabilities, AI technology is playing a key role in shifting network management from a static, reactive paradigm to a dynamic, predictive, and autonomous paradigm [7].

2.1.2 | Approaches

AI-based approaches to network management generally fall into two main categories: ML/DL and RL. We summarize each category and its typical applications below.

1. **ML/DL-based approaches**: These approaches apply classical machine learning (e.g., regression, clustering, classification) and deep neural networks to network management problems. They have been used for a wide range of tasks, including:

- **Resource allocation and optimization**: Models can efficiently allocate network and computing resources (CPU, memory, bandwidth, etc.) across physical and virtual infrastructure to meet performance goals [5].
- **Anomaly detection and security**: Supervised or unsupervised ML models train complex traffic patterns and detect abnormal states such as faults, performance degradation, or security threats. This enables early warning systems for network failures or attacks [30].
- **Data preprocessing and feature extraction**: Techniques such as dimensionality reduction and automated feature extraction help distill large-scale network data into meaningful indicators, improving the accuracy and speed of subsequent decision-making models [7, 8].

2. **RL-based approaches**: These approaches utilize RL agents that interact with the network environment to learn optimal management policies through trial and error. Notable applications include:

- **Dynamic decision making**: RL agents have been applied to problems like dynamic resource management, routing optimization (e.g., finding efficient paths or service function chains [SFCs]), and real-time traffic engineering. The agent learns policies which maximize long-term performance (throughput, latency, etc.) by continually adjusting to network feedback [31].
- **Autonomous adaptation**: An RL-based system can adapt to changing network conditions immediately. For example, the system trains to reroute traffic when congestion is detected or to balance loads after a node failure, all without explicit reprogramming. This supports self-healing and self-optimizing network behavior [5].
- **Multiagent systems**: In complex environments such as MEC or large 5G/6G networks, multiple RL agents can operate concurrently. Research in multiagent reinforcement learning explores how agents can cooperate or compete in a distributed setting to achieve global optimization and efficient resource sharing [7].

Recent research trends aim to combine the strengths of both ML/DL and RL approaches. Hybrid solutions are being explored to improve predictive accuracy while also enabling closed-loop control. For example, deep learning models might analyze traffic patterns to inform the reward function of an RL agent that manages routing decisions. In addition, solving practical network problems often involves integrating AI with other emerging technologies such as big data analytics, edge computing, and cloud-native infrastructure. These integrations help address real-time data processing needs and distributed decision-making challenges in large networks.

AI-driven network management is also supported by industry standards and collaborative initiatives. Frameworks like ETSI's zero-touch network and service management (ZSM) [32] and 3GPP's analytics functions (network function data analytics function [NFDAF] [33] and management data analytics function [MDAF] [34]) provide architectural guidelines for incorporating AI/ML into network management. Open-source projects such as open network automation platform (ONAP),[1] OSM (Open Source MANO),[2] and the O-RAN[3] Alliance are actively building platforms which embed AI for automated orchestration and management. As these technologies converge, we expect a continued focus on improving real-time responsiveness and reliability. This convergence will play a key role in fully intelligent, self-driving networks.

2.1.3 | Intent-Based Networking

Along with ML/DL and RL-based approaches, the concept of IBN has been proposed to improve and simplify network configuration, operation, and management to address the management complexity of large-scale network infrastructure [9]. The term "intent," a key keyword in IBN, refers to an abstract request and requirement for the desired state of the network, which can be expressed in natural language or expressed as letters or numbers within a predefined input format [9, 35].

Network administrators and service providers describe what the state of the network and services should be through intents but do not describe how to achieve that state. This abstraction helps reduce management complexity, minimizing incorrect network and service configurations and human errors. IBN can help reduce the time and effort required by network managers, thereby reducing the cost of operation and management [35].

An IBN system has three main tasks: intent translation, intent implementation, and continuous assurance. First, intent translation involves converting high-level intents into low-level language or network configuration instructions. Then, intent implementation automatically applies these low-level instructions to the network infrastructure. Finally, if necessary, IBN should continuously monitor and adjust the network to ensure that the network state applied with intent matches the actual intent of the network administrator and service provider.

Although IBN still has some limitations in that it should follow somewhat formalized inputs, it can express intents more freely compared with traditional methods. Network administrators can use it to perform various tasks required for network management and operation, such as network design and configuration, monitoring and orchestration, and optimization. Recently, new and effective methods for intent conversion have been proposed through recent breakthroughs in natural language understanding such as LLM. Table 2 describes a summary of existing AI-based network management approaches.

TABLE 2 | Summary of AI-based approaches for network management.

Approach	Description	Key applications	Advantages	Challenges
ML/DL-based	Using ML/DL to analyze network data, perform anomaly detection, and extract features	Resource optimization, anomaly detection, data processing	• Learns efficiently from big data—High pattern recognition accuracy—Supports early issue detection	• Complex feature engineering—Requires labeled data—Risk of overfitting
RL-based	Employs RL for dynamic decision-making and autonomous network operation	Resource management, path optimization, traffic scheduling, autonomous operation	• Real-time decision-making—Autonomous adaptation—Optimal policy derivation	• Convergence and stability issues—Scalability challenges—High training cost
IBN	Translates high-level intent into network commands for automated design, configuration, and orchestration	Network design, configuration, monitoring, and orchestration	• Simplifies configuration—Reduces human errors—Lowers operational costs	• Limited by predefined intents—Requires ongoing monitoring for dynamic changes

2.2 | Large Language Models

2.2.1 | Overview

LLMs have attracted considerable attention in industry and academia due to the introduction of ChatGPT [36]. LLMs are based on a transformer neural network architecture [12] and show high performance in processing natural language such as text and speech.

Recently, LLMs are trained on numerous Internet-scale data sets extracted from website repositories including books, articles, and programming source codes. LLMs have the ability to understand concepts from various fields and interact with humans. Typically, the size of LLMs ranges from tens of billions to hundreds of billions of parameters. With this large-scale training and wide-scale model size, LLMs can demonstrate high performance in a variety of language-based tasks, including basic question answering, document summarization, translation, text generation, and image analysis [11]. LLMs are emerging as innovative tools in various fields for language understanding and generation, and have significant potential for applications in network management and operation.

Recent research in the networking field focuses on how to apply LLMs to network environments. However, there are many challenges to applying LLMs to the networking field, and in particular, it is necessary to use network domain-specific optimized LLMs that can understand network situations. Fine-tuning and in-context learning are commonly used techniques to improve the performance of LLM on domain-specific tasks [11].

2.2.2 | Domain-Specific Adaptation Techniques

To effectively apply LLMs in network management, it is important to use domain-specific adaptation techniques to optimize the model for the network context. Primary approaches include fine-tuning, prompt engineering, Retrieval-Augmented Generation (RAG) [13], and feedback-based adaptation. Each of these techniques adjusts the LLM in a different way, and these techniques can be used complementarily to maximize the performance of the LLM.

1. **Fine-tuning** involves additional training on a pretrained LLM on domain-specific data to specialize it for tasks of the domain [37]. By continuing training on labeled examples from a specific domain, the model can achieve higher precision on tasks such as fault diagnosis or policy generation. The fine-tuned model has improved ability to understand terms and contexts of the domain and generate customized responses. Consequently, the performance for in-domain queries can be significantly improved compared with general LLMs.

However, fine-tuning requires a sufficient volume of quality training data to avoid overfitting, and it requires time and computational costs. If the model is trained too much, there can be a risk of catastrophic forgetting about the general knowledge base. To reduce these risks, parameter efficiency techniques such as Low-Rank Adaptation (LoRA) [38] and incremental layer-by-layer training, adapter modules are often used to mitigate resource costs and preserve generality. LoRA is currently attracting attention as the most resource-efficient fine-tuning method. LoRA adds low-rank matrices to the layers of the model. During fine-tuning, only these matrices are updated, and the weights of the original model remain the same.

The second approach, in-context learning, is a method which induces the model to perform a specific task by providing several examples (contexts) within the input prompt without modifying the parameters of the LLM. This method is flexible in that it can use the capabilities of the original LLM without a separate retraining process which updates the weights of the model, and can be quickly applied to various domain tasks. The user includes input–output examples necessary for problem solving in the prompt, and the LLM derives the results by referring to the shared latent concepts between the examples and the actual input. In-context learning is useful when the available dataset is small and the number of data is not sufficient for effective fine-tuning. However, the number of examples which can be included in the context is limited, making it difficult to sufficiently capture complex domain information, and the consistency of the output may be a problem because it is sensitive to the prompt design.

2. **Prompt engineering** is a strategy to effectively utilize in-context learning concepts, in which a model trains context within a prompt. This technique induces the model to perform desired tasks by designing sophisticated input prompts without updating the LLM's parameters. This approach comprises techniques such as zero-shot prompts (using only a task instruction) and few-shot prompts (including a few input–output examples as demonstrations) to steer the model. More advanced prompting methods, such as chain-of-thought (CoT) [39] prompting, facilitate the model to generate step-by-step reasoning, and prompt chaining breaks down complex tasks into a sequence of simpler prompts.

Prompt engineering allows rapid adaptation to domain tasks by using the pretrained model's knowledge without the need for time-consuming retraining. It is especially useful when only a limited number of domain-specific examples are available; LLMs can often generalize from just a few demonstrations. However, the effectiveness of this approach is highly dependent on prompt designs. Because the amount of domain information that can be provided in the prompt is limited by the length of the context, it is challenging to provide complex domain knowledge. In addition, the output can be inconsistent or suboptimal if the prompt is poorly designed. In summary, prompt engineering offers greater flexibility and immediate deployment compared with fine-tuning, but may struggle to reliably convey specialized knowledge or maintain consistency because of the prompt sensitivity.

3. **Retrieval-augmented generation (RAG)** combines a generative LLM with an external information retrieval module. This technique is useful for incorporating the latest domain knowledge in a rapidly changing field such as network management. RAG constructs a repository of domain-specific documents or a knowledge base, and when a query is given, it retrieves relevant information and provides it to the generative model. By grounding the LLM's input with up-to-date retrieved context, this approach compensates for the limitations of static pretrained models. RAG allows for real-time use of the latest external information and can improve the accuracy of responses.

However, integrating a search module with the generation module increases the system complexity: overall answer quality is highly dependent on the retrieval performance, and latency may increase because of the additional lookup process. In effect, RAG can be essential for network management tasks that require authoritative knowledge (e.g., up-to-date configuration standards or policies), but it requires maintaining an accurate and efficient search index.

4. **Feedback-based adaptation** has emerged as another powerful technique, which uses feedback signals from humans or the model itself to refine the LLM's performance. One prominent approach is RL from human feedback (RLHF) [40], where humans evaluate or rank the LLM's outputs and a reward model is trained to reflect their preferences. The LLM is then further optimized through RL to

maximize this reward, effectively fine-tuning the behavior of the model according to human-defined quality criteria. This method can greatly improve alignment with expert expectations and domain-specific requirements. Another way to use human feedback is human-in-the-loop [41], where a human is part of the LLM's decision-making processes by continuously monitoring and improving performance. In contrast, in self-refinement [42], LLM iteratively evaluates and improves its own output without external data. When the model generates an initial solution, then it evaluates its answer using the same model to identify errors or improvements, and generates a refined output in subsequent attempts.

Such feedback loops enable continuous evaluation of LLM performance during deployment. In network management scenarios, a feedback-based loop allows the model to learn from invalid commands or inefficiencies, and gradually enhance its decision-making. Compared with one-shot adaptation techniques, feedback-based methods can yield highly adaptive and reliable systems, but they also introduce additional overhead. For example, RLHF requires collecting and integrating human feedback and careful reward design, which can be resource-intensive, while self-refinement requires multiple inference iterations and careful stopping criteria. However, these methods complement fine-tuning, prompt engineering, and RAG. Feedback-based adaptation focuses on aligning and improving the behavior of the model using experiential learning, rather than expanding static knowledge or relying solely on prompt design.

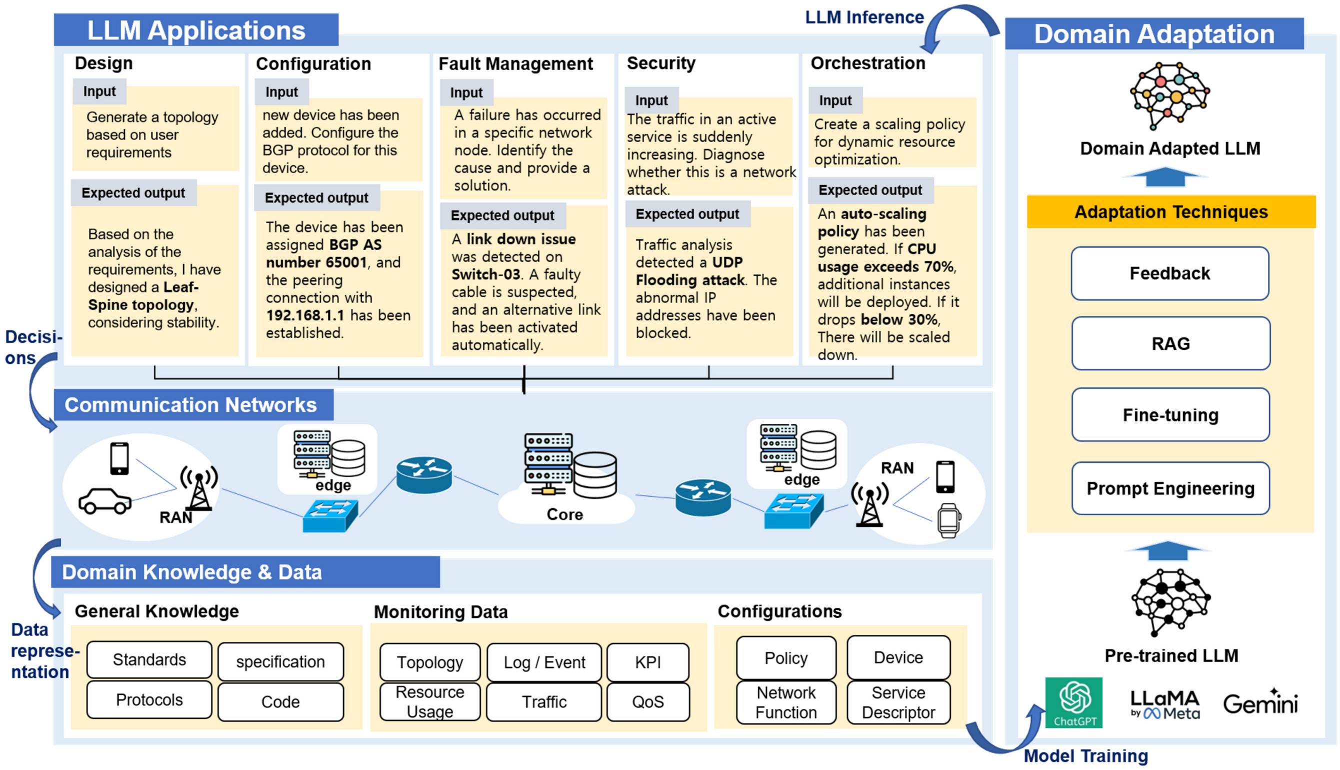
In this way, fine-tuning, prompt engineering, RAG, and feedback-based adaptation each have their own unique application methods and characteristics, and they can be used complementarily for domain adaptation and performance improvement of LLMs in network management. For example, use prompt engineering and RAG to handle general queries with up-to-date information, while using fine-tuning for specialized tasks and a feedback loop to continuously refine performance. The optimal approach may be different depending on the requirements and resource constraints of the application environment. Therefore, approaches should be carefully considered for the benefits and risks of each technique to achieve the best performance. Table 3 describes a summary of domain adaptation techniques for LLM.

TABLE 3 | Domain adaptation techniques for LLM.

Method	Definition	When needed	Implementation	Advantages	Limitations
Fine-tuning	Additional training on domain-specific data.	When high domain precision and labeled data exist.	Supervised training on network-specific corpus (full/efficient tuning).	• Improves domain accuracy and task performance.	• Data-intensive—Risk of overfitting—Time-consuming
Prompt engineering	Designing prompts (instructions/examples) without weight updates.	When rapid adaptation is required or data is insufficient.	Use zero-shot or few-shot prompts (learning), CoT, or prompt chaining.	• No retraining needed—Fast and flexible.	• Sensitive to prompt—Context limits—Inconsistency
RAG	Augmenting LLM with real-time retrieval of external knowledge.	When up-to-date or detailed info is needed.	Retrieve relevant documents from a domain knowledge base to augment input.	• Ensures current, fact-based responses—Avoids forgetting.	• Complex system—Requires maintenance—Potential latency
Feedback-based	Iteratively improving outputs using human or self-feedback.	When ongoing reliability and performance enhancement are needed.	Employ RLHF or self-refinement loops to refine outputs.	• Aligns with expert preferences—Enhances safety and correctness	• Extra steps required—Resource intensive—Design challenges

3 | LLM Adaptation for Network Management

In this section, we describe the areas in which LLM can be adapted and utilized in network fields. LLM can be used as a tool to interpret complex network data and the intent of network administrators or service providers, and support automated decision-making based on its powerful natural language understanding, generation, and inference capabilities. We classify five main tasks for network management and examine how LLM can be adapted: network design, configuration, fault management, security, and orchestration, which includes automation and optimization. Figure 2 describes the overall processes, adaptation, and applications for the network domain.



Design — Input: Generate a topology based on user requirements

Design — Expected output: Based on the analysis of the requirements, I have designed a **Leaf-Spine topology**, considering stability.

Configuration — Input: new device has been added. Configure the BGP protocol for this device.

Configuration — Expected output: The device has been assigned **BGP AS number 65001**, and the peering connection with **192.168.1.1** has been established.

Fault Management — Input: A failure has occurred in a specific network node. Identify the cause and provide a solution.

Fault Management — Expected output: A **link down issue** was detected on **Switch-03**. A faulty cable is suspected, and an alternative link has been activated automatically.

Security — Input: The traffic in an active service is suddenly increasing. Diagnose whether this is a network attack.

Security — Expected output: Traffic analysis detected a **UDP Flooding attack**. The abnormal IP addresses have been blocked.

Orchestration — Input: Create a scaling policy for dynamic resource optimization.

Orchestration — Expected output: An **auto-scaling policy** has been generated. If **CPU usage exceeds 70%**, additional instances will be deployed. If it drops **below 30%**, There will be scaled down.

FIGURE 2 | LLM Adaptation for Network Management.

3.1 | Network Design

Network design includes designing network topologies and architectures by considering various requirements of network administrators or service providers and constraints [43]. Existing network designing methods generally focus on obtaining optimal topologies and architectures within given constraints by using large amounts of previous designs, network performance indicators, and optimization algorithms [26]. For example, ML/DL-based approaches analyze network traffic patterns or equipment data to quantify cost, performance, and security requirements and then derive design plans through optimization techniques. RL-based approaches are also applied to network design problems in a way in which RL agents explore various design options in a simulation environment and train optimal policies. Those approaches have limitations in reflecting complex requirements or domain-specific knowledge because they are based on relatively standardized or regularized data and predefined objective functions.

In contrast, LLM-based approaches can interpret various requirements related to network designs and generate design recommendations with natural language understanding and generation capabilities [44]. As a result, LLM has an advantage in that it can accept the intents of administrators or service providers in natural language and present various scenarios and design options, rather than remaining limited to numerical optimization. For example, LLM receives a request such as "We need a highly scalable network design which can respond to drastic growth of service demands and expansions" from the administrator in natural language. Then LLM suggests network topology options (hub and spoke, mesh topology, etc.) and components of physical and virtual networks (network devices, network functions, etc.) based on the request, and explains the advantages, constraints, and application cases of each option. In this process, LLM understands the topologies and structure of the physical/virtual networks, and guides the design directions specifically by considering the standards, security, equipment specifications, costs, and performance requirements, and so forth. Through this, network administrators and service providers can plan and design their physical and virtual networks more easily. Also, the administrators can deploy the networks based on the network designs and plans [45].

Second, LLM integrates various knowledge in the network domain to automate the creation of initial design documents and decision support. With actual implementation cases, technical white papers, and previous design databases, LLM can automatically extract standards, protocols, and security policies that should be considered during the design process, and improve the design plans [46]. For example, for complex requirements such as "Apply the SDN/NFV technology suitable for 5G network slicing services," LLM helps to verify the validity of the design proposal by referring to related cases and standard documents, and derive the optimal solution for efficient, scalable, and resilient networks.

At last, LLM can also play an important role in design review and feedback. Although existing approaches mainly provide numerical results based on evaluation indicators, LLM comprehensively analyzes and provides feedback on the validity, security policy, scalability, and cost efficiency of the design proposal along with explanations in natural language [47, 48]. It automates expert reviews of the design proposals or analyzes natural language feedback to suggest improvements. At this time, LLMs also can be utilized as a Question Answering (QA) system for the network domain understanding. This is especially useful in iterative design work or projects involving multiple stakeholders, and improves design quality while increasing the efficiency of communications. In the future, design tools which use LLM-based approaches are expected to play an even more important role in design automation and real-time collaboration environments.

3.2 | Network Configuration

In network configuration, there exists a huge number of heterogeneous devices in the various networks, such as routers, switches, and servers. Network configurations are usually performed through a set of defined policies or commands and deal with detailed settings such as initial configurations or changes of network equipment and components [2, 7].

Existing ML/DL-based approaches focus mainly on optimizing the settings of individual network devices or components based on predefined rules and algorithms. For example, existing automation tools operate by sending configuration commands to SNMP, NETCONF, REST API, and so forth, using standardized configurations and templates. RL-based approaches train optimal routing policies through real-time interactions in a network environment, and are useful in repetitive configuration-changing scenarios. Because these approaches are based on quantitative data and preconfigured policies, they have limitations when the requirements are complex or contextually diverse [49].

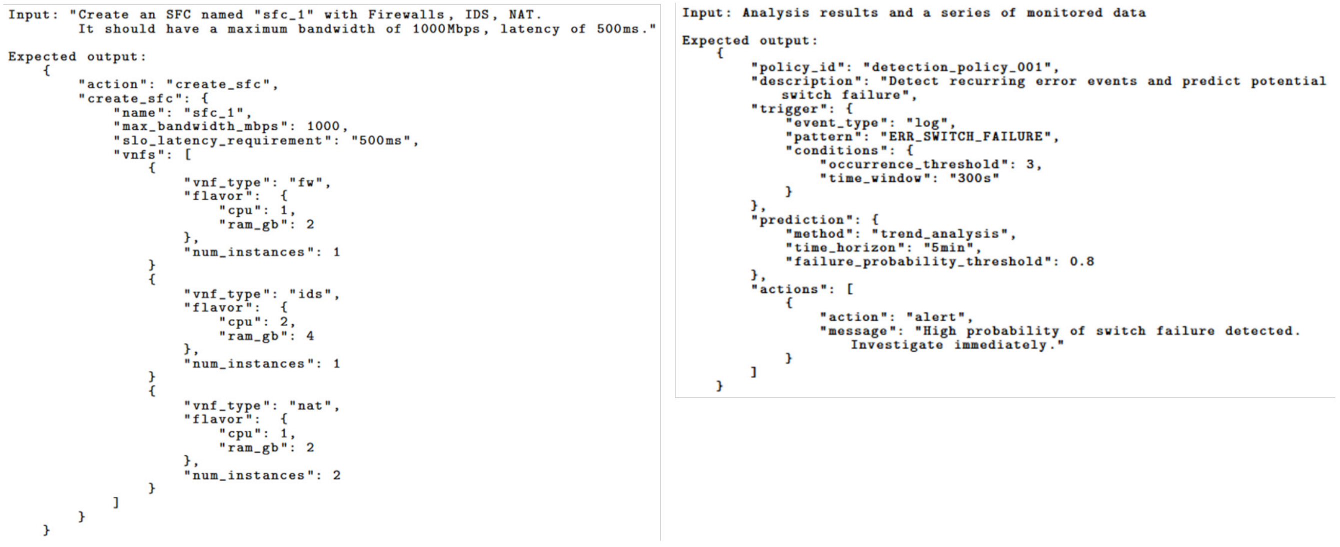
In contrast, LLM interprets the administrator's intents with natural language and translates it into specific configuration commands or scripts [50, 51]. For example, if an administrator requests “Enable BGP peering on all routers and automatically update policies,” LLM can analyze the request and derive a set of related configuration commands. This approach automates complex configuration tasks and significantly reduces human errors that may occur during manual configuration.

In routing control, LLM can be used to determine the optimal route by analyzing network topology, link status, and real-time traffic data. When the administrator makes a request such as “Configure the optimal routing path with the current network status,” LLM synthesizes past routing data and real-time monitoring data to suggest the optimal route. The administrators or systems then set the proposed configurations to their devices and provide stable network connections and services [52].

Also, in the softwarized and virtualized networks through SDN/NFV, LLM is also used for resource allocation of networks and servers, service chains, and network slice configurations [53]. LLM analyzes real-time monitoring network data and service or user requirements to optimize and allocate physical and virtual network resources (CPU, memory, bandwidth, etc.). For example, if a request such as “Allocate the resources required for each instance according to the current network and server overload status” is entered, LLM generates a resource allocation policy based on past usage patterns and real-time data and applies it to the network management system.

In the case of SFC and network slicing, LLM analyzes various service requirements and QoS/SLO (service-level objective) to efficiently divide network resources into multiple slices [26]. For example, if a request such as “Create SFC according to service requirements” is received, as shown in Figure 3a, LLM derives the optimal instances and paths to be allocated to the new SFC based on requirements and network monitoring data. Through the resource allocation process, it minimizes mutual interference between services and provides a customized network to effectively meet the demands of various users and applications. This ensures efficient resource usage and helps maintain network performance without overload or resource waste.

Finally, LLM supports automatic verification and optimization of network configuration policies. When generating a set of network configuration commands or scripts, LLM suggests optimal configuration settings by referring to previous configuration error cases or security policy violation cases [52]. This is very important for ensuring consistency and reliability



```
Input: "Create an SFC named "sfc_1" with Firewalls, IDS, NAT.
        It should have a maximum bandwidth of 1000Mbps, latency of 500ms."

Expected output:
    {
        "action": "create_sfc",
        "create_sfc": {
            "name": "sfc_1",
            "max_bandwidth_mbps": 1000,
            "slo_latency_requirement": "500ms",
            "vnfs": [
                {
                    "vnf_type": "fw",
                    "flavor":  {
                        "cpu": 1,
                        "ram_gb": 2
                    },
                    "num_instances": 1
                }
                {
                    "vnf_type": "ids",
                    "flavor":  {
                        "cpu": 2,
                        "ram_gb": 4
                    },
                    "num_instances": 1
                }
                {
                    "vnf_type": "nat",
                    "flavor":  {
                        "cpu": 1,
                        "ram_gb": 2
                    },
                    "num_instances": 2
                }
            ]
        }
    }
```

```
Input: Analysis results and a series of monitored data

Expected output:
    {
        "policy_id": "detection_policy_001",
        "description": "Detect recurring error events and predict potential
            switch failure",
        "trigger": {
            "event_type": "log",
            "pattern": "ERR_SWITCH_FAILURE",
            "conditions": {
                "occurrence_threshold": 3,
                "time_window": "300s"
            }
        },
        "prediction": {
            "method": "trend_analysis",
            "time_horizon": "5min",
            "failure_probability_threshold": 0.8
        },
        "actions": [
            {
                "action": "alert",
                "message": "High probability of switch failure detected.
                    Investigate immediately."
            }
        ]
    }
```

FIGURE 3 | Examples of input and output for (a) SFC configuration (left) and (b) policy generation for fault detection (right).

across the network and can be used as a key tool for responding to rapidly changing requirements, especially in cloud-native environments or virtual network infrastructures. In this way, LLM provides optimal configurations through requirements, real-time network data analysis, and previous pattern learning for various network configuration tasks to maximize network management efficiency.

LLM also helps to manage the history of configurations and documentation. LLM analyzes logs of changed configurations and compares the status before and after changes to generate automatic reports. For example, LLM detects configuration inconsistencies expected to occur after a network reconfiguration and provides related information so that administrators can quickly notice and take action. In this process, LLM understands various configuration file formats and standard interfaces such as SNMP and NETCONF, and performs integrated analysis and verification of configuration data.

3.3 | Fault Management

In fault management, LLM is utilized to quickly capture signs of fault occurrence, such as identifying the cause and pattern of faults, by monitoring massive text-based data such as network logs, notifications, and event records in real time. Fault management consists of the processes of monitoring, analysis, detection and prediction, and fault recovery. First, in the monitoring process, various information such as logs, notifications, event data, and performance indicators generated throughout the network is collected and preprocessed in real time [8]. LLM organizes data collected from various equipment, network components, or services such as servers into a standardized format (e.g., JSON format) and continuously updates basic status information so that abnormalities can be quickly identified. For example, error logs, CPU usage, and traffic changes generated from specific switches or routers are periodically monitored to form a dataset which can be used in the future analysis process [54].

Next, based on the collected data, LLM analyzes the patterns and correlations in detail, related to the cause of the failure [55, 56]. This process includes a comparative analysis with previous failure cases, operation manuals, and standard operating procedures (SOPs). This allows us to determine how the current abnormality is related to similar cases in the past and what type of failure it may have developed into. LLM extracts important keywords and contexts from log messages and analyzes correlations between logs. Then, it classifies the potential causes of the failure and provides meaningful insight to the administrator. This analysis function goes beyond simple error detection to enable root cause analysis by considering the interactions of multiple events occurring in complex network environments. This can reduce the time and human resources required for failure analysis and improve reliability.

LLM-based system actually detects failures and predicts the possibility of future failures based on the information derived from the analysis process and real-time data. In this process, specific failure situations, such as "ERR_SWITCH_FAILURE pattern is repeatedly occurring on a specific switch," are identified by synthesizing existing patterns and real-time abnormal signs. Furthermore, it can learn log changes over time, traffic increase patterns, and so forth, and derive prediction results, such as "given the current status, the probability of failure occurring within the next 5 minutes is 80% or higher." For example, LLM can generate the following JSON-formatted detection policy based on analysis results and real-time monitoring data inputs: Figure 3b shows a policy example where LLM detects specific error or fault patterns in the log data, predicts the possibility of failure through temporal trend analysis, and then sends an alert to the administrator.

Finally, in the fault recovery process, LLM proposes specific recovery procedures based on the detection and prediction results and prepares actions that can be automatically executed. In this process, LLM derives recovery commands and procedures such as "reboot the switch and roll back the configuration file" or "bypass the traffic of the section where the problem occurred" by referring to past recovery cases and standard operating procedures. In addition, it monitors whether the recovery actions are executed and evaluates the results after the recovery is completed, and suggests follow-up responses if additional actions are needed.

In this way, LLM-based network fault management supports the overall process from early warning before a fault occurs to automatic recovery through four stages of monitoring, analysis, detection, and prediction, and fault recovery. Existing approaches mainly extract standardized features and respond to repetitive and predictable situations through learned models. At each stage, LLM can play an important role in strengthening the resilience of the network and minimizing service downtime due to faults by more flexibly combining real-time data analysis and past case learning [57].

3.4 | Network Security

In network security, LLM analyzes data such as security logs, network traffic data, and intrusion detection system (IDS) alerts to identify threats and abnormal behaviors. For example, LLM monitors massive security log data in real-time to detect abnormal network behaviors and issue alerts so that administrators can respond immediately [7].

These detection methods, even in existing ML/DL-based studies and security solutions, extract predefined features such as large amounts of server/network logs and traffic data, and identify network attacks and potential threats through anomaly detection (classification, clustering, regression, etc.) [58]. The existing studies use models such as ML-based models (random forest, GBM, XGBoost, etc.) or DL-based models (CNN, RNN, LSTM, Transformer, etc.) to identify and classify the attacks and intrusions. In general, ML/DL-based approaches train models based on specific patterns or prelabeled data, so those methods perform with high accuracy for relatively standardized and formalized attack types. However, there are limitations in that the model's versatility is limited for complex or fast-changing attack patterns, or it is difficult to respond quickly to new threats [59].

In contrast, LLM-based approaches extract more flexible and rich contextual information from security logs or traffic data by leveraging natural language understanding and generation capabilities [60]. For example, LLM can interpret ambiguous expressions such as "abnormal login attempts" or "suspicious traffic patterns," and generate specific commands by comprehensively analyzing security policy documents or regulatory requirements, thereby identifying and responding to threats. This plays an important role in maintaining the consistency of security policies and minimizing security vulnerabilities, and can be useful in complex network environments which are linked with various security devices.

Furthermore, LLM can be utilized as a key tool in the response and analysis process after a security incident occurs. It performs the role of deriving the root cause of the incident and improving measures by integrating and analyzing security incident reports and related cases. In this way, LLM can increase the efficiency of network security operations and improve the overall security level throughout the entire network security monitoring, prevention, and incident response process.

3.5 | Network Orchestration

In network orchestration which includes automation such as self-driving networks, LLM analyzes network data across the network to perform autonomous decision-making and optimization [7]. For example, LLM collects and analyzes various sensor data, traffic monitoring data, and logs in real-time to support operational decisions such as network resource allocation, traffic distribution, and policy updates. These functions help the network maintain an optimal state on its own without the network manager having to manage the details one by one.

Compared with existing ML/DL and RL-based approaches, the LLM-based approach provides more intuitive and context-based decision support. Existing approaches support network resource allocation, traffic distribution, and routing control through numerical optimization and pattern recognition, similar to other areas of network management [61]. In addition, the RL-based approach has the advantage of real-time decision making and responding to dynamic environmental changes, as the agent interacts with the network environment and derives an optimal policy through iterative learning. However, this approach has challenges such as learning stability and search time because it is difficult to reflect unstructured data and various requirements occurring in a complex network operation environment in real time.

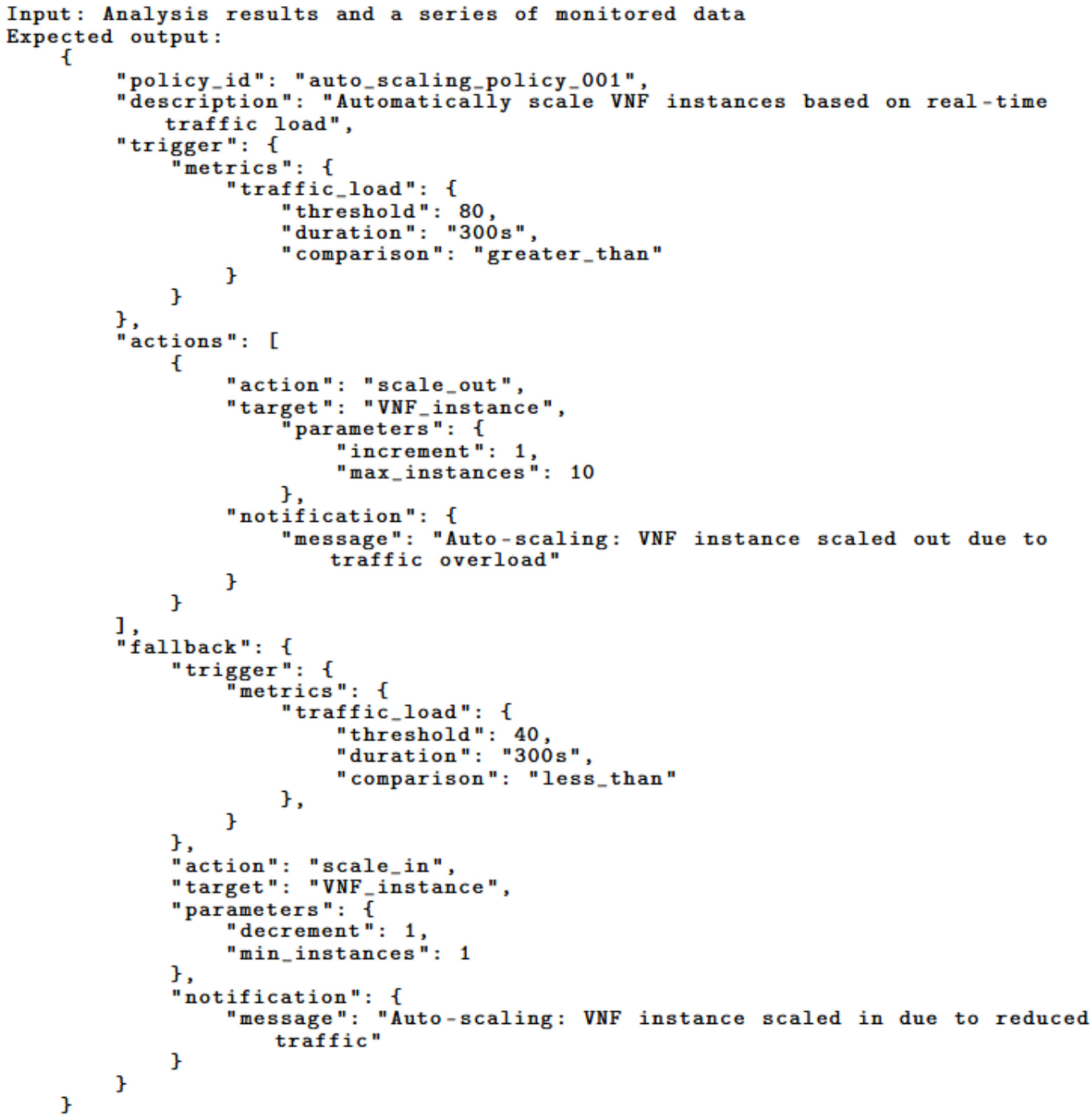
The LLM-based approach comprehensively analyzes operational data across the network from logs, sensor data, traffic monitoring data, and event records, and interprets the administrator's intent. This creates a management policy that can adapt to the network environment, which changes in real time, thereby implementing advanced orchestration. In particular, LLM shows a strength in dynamically changing physical and virtual network automation tasks. Tasks for managing networks, such as deployment, autoscaling, and migration, LLMs reallocate resources according to changing service demands and traffic loads, automatically deploy new services, and migrate VNFs to appropriate locations [62]. For example, when a new service is requested, LLM can analyze the current network status and past deployment cases to create an optimal deployment policy and script. In addition, it monitors traffic changes or resource utilization data in real time, derives an autoscaling policy, and supports decisions to increase or decrease instances, and in migration tasks, it can suggest a step-by-step execution strategy for service transfer to minimize downtime [54].

Figure 4 describes an example of generating an autoscaling policy based on input. This policy includes a condition that increases the VNF instance by 1 (scale-out) when the traffic load is over 80% for 5 min, and conversely reduces the VNF instance (scale-in) when the traffic load and CPU utilization fall below a certain level. This allows LLM to analyze real-time data and automatically make decisions to optimize resource allocation. In this way, LLM-based solutions maximize the level of automation in network operations by performing complex policy decision-making and execution processes more flexibly through contextual understanding.

In addition, network and service lifecycle management is emerging as a critical task in MEC, 5G/B5G environments [63]. Existing approaches mainly depend on predefined rules and models to manage the deployment, operation, and recovery of each service. In contrast, the LLM-based approach analyzes service requirements, various events occurring during operation, and policy change history to support automated management and optimization throughout the entire lifecycle. For example, when configuring network slicing or service chaining in a 5G network, LLM can interpret the administrator's natural language commands to automatically calculate and apply resource allocation, security policies, and traffic priorities for each slice [61]. In an MEC environment, it dynamically updates the orchestration policy according to the status of edge devices and real-time traffic changes, thereby ensuring the flexibility and stability of the network.

Finally, LLM forms a closed feedback loop in the orchestration process to support continuous network status improvement and training [41, 64]. By analyzing various events and policy changes during network operation and reflecting them in future strategies, the network will gradually operate autonomously. This approach will contribute to maximizing the network's scalability and real-time response capabilities.

Table 4 describes a summary of the LLM-based approach across the entire network management from network design to orchestration. Traditional AI-based approaches have strengths in numerical optimization such as KPIs and more structured data and rule-based network management. In contrast, LLM-based approaches have strengths in comprehensively reflecting more complex requirements and domain knowledge by utilizing natural language processing capabilities. Although the two approaches have differences, they play complementary roles and have the potential to greatly improve the flexibility and efficiency of future network design, configuration, and orchestration.



```
Input: Analysis results and a series of monitored data
Expected output:
    {
        "policy_id": "auto_scaling_policy_001",
        "description": "Automatically scale VNF instances based on real-time
            traffic load",
        "trigger": {
            "metrics": {
                "traffic_load": {
                    "threshold": 80,
                    "duration": "300s",
                    "comparison": "greater_than"
                }
            }
        },
        "actions": [
            {
                "action": "scale_out",
                "target": "VNF_instance",
                    "parameters": {
                        "increment": 1,
                        "max_instances": 10
                    },
                "notification": {
                    "message": "Auto-scaling: VNF instance scaled out due to
                        traffic overload"
                }
            }
        ],
        "fallback": {
            "trigger": {
                "metrics": {
                    "traffic_load": {
                        "threshold": 40,
                        "duration": "300s",
                        "comparison": "less_than"
                    },
                }
            },
            "action": "scale_in",
            "target": "VNF_instance",
            "parameters": {
                "decrement": 1,
                "min_instances": 1
            },
            "notification": {
                "message": "Auto-scaling: VNF instance scaled in due to reduced
                    traffic"
            }
        }
    }
```

FIGURE 4 | An example of policy generation for orchestration (autoscaling scenario).

4 | Case Studies

In this section, we explain and analyze the LLM-based network management case studies by each area in the previous section: network design, configuration, fault management, security, and orchestration.

4.1 | Network Design

In network design, most studies focus on understanding the network domain knowledge and designing networks. LLM understands network domain-specific knowledge such as topologies and protocols based on network policies, standards, and equipment requirements, analyzes the intent of network managers, and designs the networks. Furthermore, it deploys the initial state of the physical and virtual networks based on their designs.

4.1.1 | Domain Knowledge Understanding

Studies on network domain knowledge understanding evaluate LLMs to interpret and understand specialized terminologies,

TABLE 4 | Summary of LLM-based approaches in network management (summarized).

Tasks	Description	Key techniques/ examples	Key advantages	Use cases
Design	Interprets natural language requirements to suggest optimal topologies/architectures.	Natural language input interpretation, domain knowledge integration, automated design feedback	• Handles complex requirements—Automates design process	QA for network domain, network design and planning, deployment.
Configuration	Translates administrator inputs into configuration commands/scripts.	Natural language-to-code conversion, verification and documentation.	• Reduces human errors—Simplifies complex tasks	Device configurations, network configuration such as routing control, SFC, etc.
Fault management	Monitors/analyzes logs and events to detect anomalies, predict failures, and support recovery.	Real-time data collection, feature extraction, detection and recovery policy generation.	• Early anomaly detection—Root-cause analysis—Automated mitigation	Fault (anomaly) detection/prediction, mitigation, and recovery.
Security	Analyzes security logs, traffic, and IDS alerts to identify threats and optimize policies.	Use NLP with security data, interpretation of ambiguous events, policy/regulation integration.	• Flexible threat identification—Rapid response—Consistent updates	Traffic/log analysis, attack/intrusion detection, attack tools, etc.
Orchestration	Integrates operational data for autonomous decision-making and dynamic automation.	Sensor/event data analysis, natural language intent interpretation, JSON-based automation scripts.	• Real-time response—Autonomous operation—Dynamic resource optimization	Lifecycle management (deployment, autoscaling, migration, etc.).

standards, and technical knowledge within network management. These studies include extracting network protocols, question answering (QA), converting network requirements into JSON or API calls, and analyzing network metrics and KPIs. Most studies have employed BERT-based models or GPT-based models. The datasets commonly used to train and evaluate LLMs' domain understanding include 3GPP, IEEE standards, and RFC documents.

Holm et al. [48] proposed a BERT-based ELECTRA model specifically for the telecom domain QA, utilizing adaptive fine-tuning and pretraining on network QA datasets such as SQuAD and TeleQuAD. The proposed method pretrains the BERT language model on each dataset separately or combined and compares QA performance. This study was conducted with a BERT-based model (an early form of LLM), so a comparison with modern LLMs such as GPT and Llama is needed, and it only supports general QA in the network domain. However, it presents an approach for domain adaptation using an early language model and demonstrates the feasibility of network management through NLP-based QA systems in the telecommunication networks. Karapantelakis et al. [65] designed a lightweight TeleRoBERTa model with RAG to improve information retrieval and QA performance on 3GPP documents. In a QA evaluation on a 3GPP standard document dataset, TeleRoBERTa achieved 94% accuracy while using only 1/100 of the parameters of GPT-4, demonstrating the practicality of an LLM-based 3GPP document search and QA system.

The emergence of advanced LLMs such as GPT significantly enhanced the natural language understanding capabilities of QA systems. Soman et al. [47] conducted comparative evaluations of several LLMs, including GPT-3.5 and GPT-4, using prompt engineering to analyze their performance and limitations specifically within the telecom domain. They apply prompt engineering to evaluate domain adaptation, context retention, hallucination, response reliability, and perturbation robustness in the telecom domain. However, this study did not use domain optimization techniques and used a private dataset. Nonetheless, by learning network domain knowledge with LLMs, it suggests the possibility of supporting network and equipment management and troubleshooting, aiding network administrators. Similarly, Zou et al. [66] proposed TelecomGPT, fine-tuned on the telecom-specific OpenTelecom dataset.

Kotaru [67] developed the DIO (Data Intelligence for Operators) Copilot system, which applies few-shot learning and feedback mechanisms using GPT-4 to query and analyze 5G network metrics and KPIs through natural language. In a 5G core network environment, they measured query execution accuracy and found that DIO Copilot achieved 18% higher execution accuracy than an existing natural language to SQL system (DIN-SQL). DIO Copilot still requires expert intervention in real-time operations and does not account for heterogeneous vendor data formats. However, it supports network operators with natural language-based data retrieval and analysis, and introduces a continuous improvement mechanism through expert feedback.

Beyond QA systems, other studies have focused on understanding and analyzing standards such as 3GPP and IEEE, aiming to improve retrieval and summarization performance. Bariah et al. [68] fine-tuned BERT-based models to achieve 84.6% accuracy for classifying 3GPP technical documents. This study did not include real network data beyond 3GPP technical documents, and the optimization of the GPT-2 model was relatively low. However, it demonstrates that LLMs can effectively understand technical documents in the telecom domain and suggests their potential use for network automation in telecommunications. Sharma and Yegneswaran [46] proposed the PROSPER framework, using GPT-3.5 with zero/few-shot learning to automatically extract network protocol states from RFC documents. This study demonstrates that PROSPER achieves 1.3 times higher accuracy and 6.5 times lower false positive rate than existing FSM extraction models.

Similarly, Bornea et al. [69] used prompt engineering and RAG techniques with multiple LLMs (GPT, LLaMA, Mistral) to optimize the retrieval performance on 3GPP standards through their Telco-RAG system. Telco-RAG optimizes RAG with models GPT-4, GPT-3.5, LLaMA-2, and Mistral, improving search accuracy by 6.6%–14.45%, reducing RAM usage by 45%, and increasing processing speed by over 50× compared with a standard RAG. This study's strength is that it optimizes the RAG architecture for more reliable search and information processing compared with existing LLM-based search systems. And Shao et al. [55] designed WirelessLLM, which enhances retrieval and summarization capabilities for 3GPP and IEEE 802.11 documents by integrating GPT-4 and Claude 3 Opus with prompt engineering, fine-tuning, and RAG techniques. While WirelessLLM still exhibits some hallucination issues, it improved 3GPP protocol document understanding accuracy by 13.45% over GPT-4, demonstrating the potential of applying LLMs for wireless network optimization. These studies have shown potential in 3GPP and IEEE document understanding, however, they still have some issues such as hallucinations, knowledge bias, and contextual misunderstanding.

Others address interpreting network requirements in natural language, extracting user intents, and assessing reproducibility. Wang et al. [70] proposed the NetConfEval framework, which employs fine-tuned GPT-3.5/4 and Code LLaMA models to translate network requirements into JSON and API calls. However, if LLMs do not learn enough prior knowledge specific to the network domain, there is a possibility that interpretation errors will occur for some network components (e.g., specific SDN API calls), and it is difficult to guarantee consistent conversion results in complex multivendor environments.

Manias et al. [71] applied GPT-3.5 with prompt engineering to classify and extract user intents from 3GPP-based network requests. The proposed method uses GPT-3.5 to verify that the LLM can accurately extract both single and composite intents and discusses the potential integration of open-source models and 5G networks. However, the proposed method requires additional fine-tuning for ambiguous requests and a proof-of-concept for validation in actual network environments. Lastly, Xiang et al. [72] evaluated the automation and reproducibility of network-related research papers (e.g., NCFlow [73], APKeep [74], Arrow [75], APsystem [76]) using prompt-engineered

ChatGPT. However, the proposed method shows a high error rate and increased execution time due to the lack of prompt engineering and performance optimization.

In summary, these studies took advantage of the fact that LLM-based approaches handle complex, diverse, and natural language-based data more effectively than traditional rule-based or manual methods. In particular, LLMs excel at flexible and automated execution of complicated tasks such as protocol information extraction and converting natural language inputs into network APIs and JSON formats.

However, common limitations across these studies include reliability issues such as hallucinations, where the models produce inaccurate or inconsistent outputs, especially when limited domain-specific data. Most of the domain knowledge understanding studies investigated use limited datasets or prompt engineering such as few-shot learning. However, using RAG, which uses sophisticated techniques such as filtering [77] or an iterative feedback-based mechanism to generate better output such as self-refinement [42] can reduce hallucinations and further improve QA performance in the network domain.

4.1.2 | Design and Planning

Studies on network designs and planning focused on using LLMs to automate the interpretation of network topologies, design optimization, and translating network administrators' high-level intents into practical designs. Key tasks for design and planning include network topology diagrams, network blueprints, and translating user intents into network service descriptors (NSDs).

Several studies focused specifically on network topology analysis and design. Ifland et al. [78] propose GeNet, using GPT-4 to interpret topology diagrams and generate network configurations based on natural language requirements. The authors measure the topology understanding of the proposed method in a GNS3Vault simulation environment and provide text-based topology design. The method facilitates network design planning and enhances managers' decision support, but it has limitations in handling ambiguous or complex requirements.

Donadel et al. [45] evaluated the capabilities and limitations of various LLMs (GPT-4, Copilot, LLaMA 2, Mistral) for analyzing network topologies, noting particular difficulties in understanding more complex network structures. The authors show the GPT-4-based Copilot performed the best average accuracy with 79.3%. Similarly, Ayed et al. [79] proposed Hermes, combining GPT-4o and LLaMA 3.1 with a digital twin methodology to translate network policies into deployment blueprints. The proposed methods show high understanding in small and simple networks; however, the performance reduces as the network complexity increases, and the models are not reliable in large networks.

Huang et al. [80] proposed ChatNet, integrating GPT-4 with optimization tools (Cplex, NetworkX) and RAG to optimize network designs for efficient traffic management and resource utilization. ChatNet improves network design accuracy by 17% when applying RAG and prompt engineering, and improves optimization speed compared with existing manual design methods. The proposed method has high computational and learning costs, so model weight reduction and verification in real environments are necessary, and it is still difficult to replace expert-level decision-making.

Other studies targeted specialized modeling tasks such as NSD generation and network planning. Jiang et al. [81] presented CommLLM, a GPT-3.5-based multiagent framework designed for autonomous extraction, iterative planning, and placement in communication networks. However, because this study does not discuss the slow response time of multiagent systems, it needs to be verified for real-world application. Quan et al. [82] proposed an automated framework using GPT-4 and Claude for generating radio maps and wireless network planning, significantly reducing manual effort. Mekrache and Ksentini [83] used Code LLaMA with few-shot learning to convert high-level user intents into ETSI-compliant NSDs, with the network deployment process. The authors validate the proposed method on the EURECOM 5G testbed, and the Code LLaMA 34B model shows an NSD conversion accuracy of 93.4% and a verification success rate of 97.8%. However, due to the latency of the LLM, a bottleneck occurs as the number of NSDs increases.

Most studies have used GPT or LLaMA-based models trained on datasets collected from simulated environments such as GNS3Vault, ETSI standards, or custom digital twin frameworks [78, 79]. These studies have demonstrated that LLM-based approaches can improve the automation and flexibility of network design tasks compared with manual or rule-based methods. In particular, these approaches show strength in interpreting high-level intents and generating initial topology suggestions.

However, several limitations still remain. First, most studies focus on partial stages of the design workflow such as intent interpretation or blueprint generation, and lack integrated end-to-end pipelines that include verification and deployment [79, 80]. Second, LLMs often struggle to translate ambiguous or complex intents, especially in heterogeneous or large-scale environments, where semantic inconsistencies or hallucinations may occur frequently [81, 82]. Third, the high computational requirements and inference latency of LLMs limit their applicability in dynamic network environments that require real-time responsiveness [80, 83]. In addition, many models have focused on specific data or conditions, making it difficult to generalize their approaches to diverse real-world network environments.

To address these limitations, future research should explore lightweight LLM architectures and domain-specific adaptation techniques which improve reasoning consistency [84]. It is also important to incorporate simulation-based or digital twin-based validation procedures before deployment, in order to reduce the risks of hallucination and security vulnerabilities and to improve reliability in operational environments [41, 85].

4.1.3 | Deployment

Existing case studies on deployment mainly focus on the use of LLMs to automate and optimize the practical deployment of VNFs and network services. Existing studies include wireless network optimization and deployment of VNFs. LLM-based methods are combined with datasets often sourced from urban network models, virtualized testbeds, and standardized simulation environments.

Several studies focused on deploying network functions and automating network virtual environments. Marques et al. [86] used GPT-3.5 with few-shot learning to deploy VMs and create network topologies from natural language inputs, streamlining virtual network setups significantly. The proposed method compares the Single Conversation Approach and the Step-by-Step Query Approach and shows that the Step-by-Step Query Approach can build a more accurate environment by reducing the error rate for VM settings to 10%. However, the proposed method conducts simple creation commands, so complex scenarios were not conducted and cannot apply network attributes such as DHCP and NAT.

Dzeparoska et al. [87] proposed Emergence, a framework based on GPT-3.5 and GPT-4, designed for deployment, policy generation, and execution of network functions within OpenStack environments. While the existing methods operate based on a predefined set of policies and face flexibility issues for new intents, this study supports generalization to new intents by utilizing LLM. However, there is a bottleneck concern due to LLM latency, and an additional verification mechanism is needed to detect and correct execution errors of the generated policy in advance. Similarly, Ghasemirahni et al. [62] proposed FlowMage, employing GPT models (GPT-4, Code LLaMA, Gemini) to analyze code structures of stateful network functions and optimize receive-side scaling (RSS) configurations for enhancing QoS.

Other studies focused on wireless network deployments. Sevim et al. [88] presented a LLM-based RL framework aimed at optimizing wireless network deployment in urban environments, resulting in improved signal strength and reduced learning time. The authors conducted experiments in a real urban environment (Munich model) using the Sionna library and demonstrated that the CNN-LLM combined model improves the average signal reception strength by 10 dB and reduces the learning speed by 30% compared with the baseline CNN model.

Wang et al. [89] proposed an LLM-based framework integrating GPT-3.5, GPT-4, and GPT-4o models combined with RAG to optimize the deployment of base station sites, achieving enhanced accuracy and operational efficiency. These two studies show the possibility that the application of LLM improves deployment performance through methods proposed in wireless networks. However, each experiment was performed only with a specific dataset in a simulation environment, and there is a lack of discussion about overheads such as increased response time due to LLMs when applied to a real environment.

Existing studies have proposed various ways to use LLMs for deploying network functions and services, but most approaches are limited to specific tasks such as generating configuration scripts or analyzing deployment parameters [86, 88]. They often do not cover the entire deployment pipeline, including verification, integration with orchestration platforms, and real-time operation in complex network environments. Moreover, because many studies are conducted in simulation environments [89], they need to be validated in real-world network environments.

Another important challenge is the latency introduced during policy generation and execution, which can reduce responsiveness in time-sensitive deployment scenarios. Hallucinations and semantic inconsistencies in generated configurations or scripts can pose serious risks, particularly when proper validation and recovery mechanisms do not exist [90]. In addition, the security of LLM-driven deployments is a concern due to possible adversarial prompts or unintended model behaviors, and there are risks related to unauthorized access or control of network resources [91, 92]. Most models are trained on limited datasets and are tailored to specific network settings, which limit their generalizability across vendors and dynamic service environments.

To overcome these limitations, future work should develop modular LLM pipelines which separate task planning, policy generation, validation, and execution [93]. Pre-deployment verification using digital twin environments should be incorporated to reduce the risks of hallucination and ensure secure deployment [92]. It is also important to explore lightweight LLM architectures which reduce latency and resource consumption [11]. These efforts will improve the safety, scalability, and practical applicability of LLM-based deployment systems. Table 5 describes a detailed summary of the case studies related to network designs.

4.2 | Configuration

In network configuration, most studies focused on configuring various heterogeneous devices and networks. These studies usually convert high-level natural language intents to low-level device commands and network configurations.

4.2.1 | Device Configurations

Case studies on device configurations use LLMs to automate and enhance network device configurations such as routers and switches. Typical tasks include automatic generation of configuration codes, natural language translation of user intents into configurations, error detection, and verification. The majority of these studies use GPT-based models (GPT-3.5, GPT-4, GPT-4o) or Code Llama, adapting techniques (e.g., prompt engineering, fine-tuning, RAG, and feedback-based approaches). Evaluation scenarios frequently involve simulation and testbed environments, using datasets derived from simulation environments such as GNS3[4] and Kathará,[5] and manufacturer platforms such as Cisco and Juniper.

Several studies have specifically targeted automating network configuration code generation and intent translation. Wang et al. [70] proposed NetConfEval, using GPT-4 combined with RAG to automate the generation of routing and protocol configurations, achieving accuracy improvements. The proposed

TABLE 5 | Summary of case studies for network management: Design.

Ref.	Tasks	Proposed	Models	Adaptation techniques	Dataset/ evaluation
Holm et al. [48] (2021)	Network domain knowledge understanding, QA	ELECTRA, LLM-based telecom domain QA model and suggest a network domain adaptation	BERT	Adaptive fine-tuning, pretraining	Network domain QA dataset (SQuAD, TeleQuAD)
Soman et al. [47] (2023)	Telecom domain knowledge understanding, QA	Compare the QA performance and limitations of LLM models in telecom domain	GPT-3.5, GPT-4, Bard, LLaMA	Prompt engineering	Custom QA dataset based on Cradlepoint documents
Xiang et al. [72] (2023)	Reproduction of network research	Automate the reproduction of network research by using ChatGPT to replicate systems	Free ChatGPT, GPT-3.5	Prompt engineering	Reproduction accuracy (NCFlow, APKeep, Arrow, AP system)
Sharma and Yegneswaran [46] (2023)	Domain Knowledge Understanding, Protocol information extraction	PROSPER, LLM-based framework for protocol state machine extraction	GPT-3.5-Turbo	Zero/ few-shot learning	RFC documents (RFC 793, RFC 4340, RFC 2637)
Kotaru [67] (2023)	Network metrics and KPIs analysis	DIO Copilot, query system for network metric and KPI analysis	GPT-4	Few-shot learning, Expert feedback	Metric data collected from a 5G core network environment
Bariah et al. [68] (2023)	Document classification	LLM-based 3GPP technical document classification with fine-tuning	GPT-2, BERT, DistilBERT, RoBERTa	Fine-tuning	Classification accuracy on 3GPP technical documents
Wang et al. [70] (2024)	Formatting network requirements to JSON format	NetConfEval, analyze how LLMs can interpret natural language network requirements and convert them into JSON formats or network automation API calls.	GPT-4, GPT-3.5, CodeLLaMA	Fine-tuning (provided by OpenAI), QLoRA	Accuracy of converting network requirements and API calls

(Continues)

TABLE 5 | (Continued)

Ref.	Tasks	Proposed	Models	Adaptation techniques	Dataset/ evaluation
Bornea et al. [69] (2024)	Query augmentation, optimized document retrieval	Telco-RAG, Optimize the RAG architecture to enhance search accuracy	GPT-4, GPT-3.5, LLaMA 2, Mistral	Prompt engineering, RAG	3GPP standard documents dataset/accuracy, resource usage, and processing speed
Karapantelakis et al. [65] (2024)	3GPP document understanding, QA	TeleRoBERTa, LLM and RAG based QA model for 3GPP documents	TeleRoBERTa (RoBERTa-based)	RAG	QA dataset based on 3GPP standard documents
Manias et al. [71] (2024)	Intent extraction and classification	LLM-based intent extraction model based on 3GPP standards	GPT-3.5	Prompt engineering (including few-shot learning)	Extraction accuracy from 3GPP-based network request data
Shao et al. [55] (2024)	Domain knowledge understanding, document summarization	WirelessLLM, improve retrieval and summarization of standard documents (3GPP and IEEE 802.11)	GPT-4, Cloude 3 Opus	Prompt engineering, RAG, Fine-tuning	Protocol document understanding using 3GPP and IEEE 802.11 standards
Zou et al. [66] (2024)	Domain knowledge understanding, QA, document classification	TelecomGPT for Telecom domain based on LLM with fine-tuning	Llama 2, Llama 3, Mistral	Fine-tuning (QLoRA, FSDP)	Tasks for Telecom QA, document classification, code generation
Ifland et al. [78] (2024)	Intent analysis, topology design	GeNet, facilitates network design planning and decision support	GPT-4	CoT	GNS3Vault simulation environment/ Topology understanding and providing text-based topology designs

(Continues)

TABLE 5 | (Continued)

Ref.	Tasks	Proposed	Models	Adaptation techniques	Dataset/ evaluation
Donadel et al. [45] (2024)	Topology analysis, subnet identification, connectivity understanding	LLM-based network design assist framework with different network sizes	GPT-4, Copilot, Llama 2, Mistral	Prompt engineering	Network-related tasks across different network sizes
Ayed et al. [79] (2024)	Network design, code generation	Hermes, automatically generate network digital twin blueprints and convert policies into code	GPT-4o, LLaMA 3.1	CoT, self-refinement	Deployment success rate in base station placement
Mekrache and Ksentini [83] (2024)	Intent to network service descriptor conversion	Transform user intents into a NSD based on few-shot learning	Code LLaMA	Few-shot Learning, feedback (human-in-the-loop)	Conversion accuracy with EURECOM 5G testbed
Jiang et al. [81] (2024)	Domain knowledge understanding, QA, system and network modeling	CommLLM, LLM-basd QA and task execution framework for 6G networks	GPT-3.5	CoT, RAG, self-refinement, RL from evaluation	6G related research paper dataset, semantic communication model evaluation
Huang et al. [80] (2025)	Topology design	ChatNet, design and generate optimal network topologies using LLM and RAG	GPT-4 with Cplex/NetworkX	CoT, RAG	Design accuracy in network traffic and capacity allocation scenarios
Quan et al. [82] (2025)	Map generation, wireless network design	Automate radio map generation and wireless network design/planning based on task objectives and constraints	GPT-4, Claude	Prompt engineering	Signal strengths and SINR in a simulated urban environment

(Continues)

TABLE 5 | (Continued)

Ref.	Tasks	Proposed	Models	Adaptation techniques	Dataset/ evaluation
Marques et al. [86] (2023)	VM deployment, topology generation	LLM-based VMs deployment and settings, and generate network topologies from natural language inputs	GPT-3.5	Few-shot learning	Command error rates during network interface configuration and IP address assignment in a virtual environment
Dzeparoska et al. [87] (2023)	NF and service chain deployment	Emergence, LLM-based deployment, policy generation, and execution framework	GPT-3.5, GPT-4	Few-shot learning	OpenStack-based SAVI testbed with service chains
Sevim et al. [88] (2024)	Wireless network deployment	LLM-based RL framework for wireless network deployment	DistilBERT with RL (DDPG) and CNN	Prompt engineering	Signal reception rate and learning time in Sionna library
Ghasemirahni et al. [62] (2024)	Code analysis, stateful network functions (NFs) deployment	FlowMage, analyze the code structure of stateful NFs and configure RSS settings	GPT-3.5, GPT-4, Code Llama, Gemini	Prompt engineering	Throughputs in NF chain deployment scenario
Wang et al. [89] (2024)	Base station site deployment	LLM- and RAG-based BSS deployment optimization framework	GPT-3.5, GPT-4, GPT-4o	Prompt engineering, RAG	City grid dataset of coordinates and traffic

method analyzes network topology with LLM, generates optimal routing algorithms as Python code, and aims to automatically produce low-level configurations for existing and new protocols (BGP, OSPF, etc.). Experimental results show that GPT-4 achieved a 92.3% success rate in routing code generation, and an 88.4% accuracy in low-level configuration generation when using RAG approach. However, in some complex network scenarios, the generated codes had unexpected side effects (e.g., performance degradation under certain traffic patterns), and the generated configuration could conflict with network security policies.

Similarly, Wang et al. [94] presented NETBUDDY, an LLM-based system translating natural language network policies into structured configurations to accelerate and enhance reliability. Li et al. [95] proposed PreConfig, a multitask GPT-based model designed to improve scalability and accuracy in configuration translation and generation tasks.

Wei et al. [96] designed an intent-based configuration framework using GPT-4o and RAG to optimize operational efficiency when network devices change. The authors present IRAG (Intent-based RAG) to ensure consistency in in-context learning. The proposed method shows the feasibility of configuration translation across heterogeneous network devices by achieving an accuracy of 97.74% in cross-vendor device configuration translation. However, while the method can detect and correct syntax errors, it remains vulnerable to semantic errors caused by hallucinations. Additionally, this approach has limitations due to its reliance on training data and the difficulty of real-time processing stemming from the multistage LLM invocations.

Other studies focused on configuration error detection, validation, and correction. Mondal et al. [97] proposed Verified Prompt Programming (VPP), using GPT-4 with human-in-the-loop feedback to effectively identify and reduce router configuration errors. In the experiments, the LLM alone approach showed many syntax and policy errors, but the proposed approach showed better accuracy by the feedback mechanism. However, this study also discusses repeated semantic errors and correction performance issues because of the semantic hallucinations, which make the effectiveness of the proposed approach highly dependent on the capabilities of the verifier. Furthermore, the reliability of the proposed method in handling large-scale or complex configurations still remains limited.

Jeong et al. [98] presented S-Witch to automate generating CLI configurations of network devices using GPT-3.5-Turbo with CoT in a simulated digital twin environment (using GNS3). The proposed method evaluates CLI command execution success rate and configuration accuracy in a GNS3-based digital twin environment under OSPF, VLAN, and subnet configuration scenarios. S-Witch achieved a 92.3% CLI execution success rate in a small network environment, but showed relatively high error rates for complex configurations such as VLAN and OSPF. Similarly, GeNet [78] provided configuration modifications through GPT-4 with prompt engineering (CoT). Additionally, ChatNet [80] theoretically proposed automated generation methods for ACL and CLI configurations using adaptation techniques (fine-tuning, prompt engineering, and RAG) to enhance device consistency.

In summary, these LLM-based approaches demonstrate potential in translating complex device configuration tasks by reducing manual effort and improving consistency compared with traditional or rule-based methods. These methods utilize the language understanding capability of LLMs to interpret natural language intents and translate them into structured configuration commands across various vendors and protocols.

However, several studies report common challenges, including semantic inconsistencies due to hallucinations, limited domain-specific training data, and high computational requirements. Future research should focus on improving the semantic reliability of generated configurations, constructing diverse and domain-rich datasets for heterogeneous devices, and optimizing lightweight and scalable LLM architectures while considering the processing time of LLM-based configuration tasks in real-world environments.

4.2.2 | Network Configurations

Case studies on network configurations use LLMs to automate the configuration management tasks across entire network infrastructures, such as intent-based network configurations, policy implementations, and automated verification. Common tasks include generating network-wide configurations, translating high-level intents into low-level network configurations, and verification. Most of the case studies use GPT-based models (mostly GPT-4, GPT-3.5), LLaMA-based models, and Mistral, with adaptation techniques such as prompt engineering, fine-tuning, RAG, and feedback mechanisms. Evaluation environments typically involve simulated or virtual network scenarios using testbeds based on network simulators and emulators.

Most of the case studies have targeted translating high-level intents or requirements in natural language and generating low-level network configurations. Chakraborty et al. [51] proposed an AI system that converts tariff plan intents into API-based network configurations. The proposed method used GPT-3.5 Turbo and Llama 2 with prompt engineering, fine-tuning, and a human feedback mechanism. The experiment showed that the translation accuracy is about 96.15%; however, there still remain issues with hallucination of complex intent parsing and exceptional conditions. So the scope of intent parsing needs to be expanded to include policy-based templates and conditional sentence processing.

Fuad et al. [49] presented an IBN framework which converts natural language intents into BGP and firewall configurations using GPT-4/3.5, LLaMA 2, and Mistral, with few-shot learning. The authors evaluate their approach using a few representative intent examples and assess the success of configuration generation. However, this study is limited by its small-scale, handcrafted evaluation dataset, making it difficult to measure performance. Also, hallucination still remains despite prompt tuning, leading to occasional semantic errors. Similarly, Kou et al. [99] used GPT-based models to abstract natural language intents and generate graph-based network intent structures to support intent-driven network configurations.

There exist case studies for configurations in the NFV environment. Brodimas et al. [100] presented a network management framework using LLMs to translate multimodal intents into standardized configurations (NEST JSON). The proposed method showed that LLMs can translate user intents into deployable configurations. However, this study is limited to predefined intents and slice templates, and quantitative metrics such as accuracy and processing time need to be evaluated.

Similarly, Lin et al. [101] designed the AppleSeed framework, which converts natural language intents into Python codes related to service chain configurations in a cloud environment. Bandara et al. [53] proposed SliceGPT, using GPT-3.5-Turbo with blockchain to translate dynamic network slice configuration management in 5G environments. Lastly, Tu et al. [24, 102] converted natural language network intents into low-level network configuration commands using LoRA, dynamic in-context learning, and continual learning techniques by comparing various LLM models.

Other studies focus on verifying and benchmarking network configurations. Lian et al. [50] proposed the Ciri framework to detect configuration errors and provide analysis through multi-LLM voting with few-shot learning and RAG. The experiments show that Ciri achieves up to 45% higher configuration error detection rates compared with existing ML-based verification methods. However, this study has limitations in detecting errors due to dependencies between configurations and handling version-specific settings, but it demonstrates the potential usefulness of LLMs in software configuration verification.

Lira et al. [103] proposed LLM-NetCFG, which generates configurations from natural language inputs and verifies the generated configurations in a ZSM environment. LLM-NetCFG classifies network configuration intents with 92.2% accuracy, taking 5–7 min for simple configuration change requests and over 10 min for complex configurations (e.g., OSPF routing or tunnel setup). This study shows the potential of using LLMs for network configuration automation in a ZSM environment, although further validation is needed in real-world environments due to the processing time and additional tasks required to reduce errors caused by hallucination.

Similarly, Ye et al. [104] presented LLMSecConfig to detect and fix configuration errors using prompt engineering and RAG in the Kubernetes environment. Lastly, Aykurt et al. [52] implemented NetLLMBench, a benchmarking framework using a Kathará emulator to evaluate LLMs for network configuration tasks, identifying top models in both accuracy and speed.

In summary, LLM-based approaches show significant promise in translating natural language intents into executable network configurations across diverse infrastructure environments. These methods contribute to automating policy enforcement, routing, firewall management, and network slicing. However, many studies are based on small-scale and handcrafted datasets, and in some cases do not include standardized benchmarks for evaluating key performance metrics such as accuracy, processing time, and scalability [49, 100]. Hallucination and semantic inconsistency remain frequent issues when handling complex or ambiguous intents, and effective mitigation strategies have yet to be established.

Further research should address not only hallucination but also the processing time required for real-time application in operational networks. It is also important to consider how to manage security risks [105] when LLMs are used to access or modify network configurations. Finally, ensuring the reliability and explainability of generated outputs remains a critical requirement for deploying LLM-based systems in mission-critical environments. Table 6 describes the detailed summary of the case studies related to configurations.

4.3 | Fault Management

In fault management, most of the existing studies collect and analyze various network data such as logs, notifications, and network metrics in real time to diagnose whether a network fault or failure has occurred, analyze root causes, and suggest recovery procedures. Recent LLM-based approaches perform anomaly detection and fault diagnosis using metrics and logs based on LLM's network domain knowledge and natural language understanding. There are attempts to improve accuracy in fault management tasks such as root-cause analysis, mitigation, and recovery by detecting abnormal states of the network.

First, several case studies present LLM-based fault diagnosis and anomaly detection methods using network metrics. Shao et al. [55] analyzed frequency patterns in cognitive radio networks using WirelessLLM to achieve better anomaly detection performance than existing approaches. WirelessLLM also presents theoretical methods for fault source detection and automatic recovery. The proposed method showed a detection accuracy of 91%, but has limitations such as output reliability and hallucination issues due to the lack of wireless domain-specific data and difficulty in real-time application due to the processing time of LLM. Similarly, Dharmalingam et al. [106] proposed a method optimized with SFT (Supervised Fine-Tuning) and RLHF techniques by combining small LLM (OPT (Open Pretrained Transformer) model-based anomaly detection) and large LLM (Llama2-based prediction) through the Aero-LLM framework in a unmanned aerial vehicle (UAV) environment and present a method for rapid fault detection and prediction through response.

And Tang et al. [57] proposed multiscale semanticized anomaly detection model (MSADM) to apply CoT to GPT-4 and LLaMA 2 with the LSTM model to detect anomalies and diagnose faults for multiscale data. MSADM evaluated on data collected from seven network environments in ns-3[6] achieved 89% fault diagnosis accuracy. MSADM also presented a method to generate fault reports and recovery scripts. However, this study does not discuss any safety mechanisms to address potential issues such as code errors, security vulnerabilities, or unintended network damage caused by the use of LLMs. In addition, since the detection accuracy is measured based on data collected from a simulation environment, further analysis is needed regarding the

TABLE 6 | Summary of case studies for network management: Configuration.

Ref.	Tasks	Proposed	Models	Adaptation techniques	Dataset/ evaluation
Mondal et al. [97] (2023)	Generate router configurations	Verified Prompt Programming (VPP) to reduce syntax and policy error on network devices	GPT-4	Prompt engineering (VPP)	Router configuration tasks with error reduction measured against standard syntax and policy benchmarks
Wang et al. [94] (2023)	Network policy interpretation, generate high/ low-level configurations	NETBUDDY, framework for reducing configuration complexity and generate configurations	GPT-4	CoT, feedback	P4-based MPLS configurations and BGP routing policy changes with network emulator
Wang et al. [70] (2024)	Device configurations, generate routing code	LLM-based generate network routing code and low-level device configurations	GPT-4, GPT-4-Turbo	Prompt engineering, RAG, feedback	Code generation tests for routing codes and device configurations
GeNet [78] (2024)	Modify device configurations	GeNet, generate device configurations using LLM	GPT-4	CoT	Accuracy and practical applicability of generated configurations
Li et al. [95] (2024)	Configuration generation, translation, and analysis	PreConfig, configuration generation and translation using LLM	PLBART (GPT-3.5, GPT-4 for baseline models)	Prompt engineering, fine-tuning, task-adaptive pretraining	Scores for configuration generation, analysis, translation
Jeong et al. [98] (2024)	Device configurations, generate CLI commands	S-Witch, LLM-based device commands generation	GPT-3.5-Turbo	CoT	Execution success rate in digital twin environment of GNS3

(Continues)

TABLE 6 | (Continued)

Ref.	Tasks	Proposed	Models	Adaptation techniques	Dataset/ evaluation
Wei et al. [96] (2025)	Generate device configuration files, convert intents to configurations	Intent-based configuration framework for network devices using RAG and in-context learning	GPT-4o, Qwen-Max	In-context learning, RAG	Correctness and tree matching rate using router configuration data with Nokia and Huawei
ChatNet [80] (2025)	Intent translation to low-level device configurations	ChatNet-based device configuration methods using LLM and adaptation techniques	Not specified	Prompt engineering, fine-tuning, RAG	Propose theoretical methods, not evaluated
Chakraborty et al. [51] (2024)	Convert tariff plan and network policy into network configurations	AI/LLM-based key extraction and configuration generation	GPT-3.5-Turbo, LLaMA 2	Few-shot learning, fine-tuning	Accuracies for configuration generation using network service and tariff plan data of telcos
Fuad et al. [49] (2024)	Intent translation, generate network configurations	IBN framework to convert intents into network configurations using LLM	GPT-4, GPT-3.5, LLaMA 2, Mistral	Few-shot learning	BGP router, firewall configuration test with FRRouting protocol
Lian et al. [50] (2024)	Verify configurations by detecting errors	Ciri, configuration verification and error detection framework using multi-LLM voting	GPT (4, 3.5), Cloude (3-Opus, 3-Sonnet), CodeLLaMA, DeepSeek	Prompt engineering	Error detection rates with configuration dataset 10 open-source systems
Lira et al. [103] (2024)	Intent classification and translation, generate configurations	LLM-NetCFG, configuration generation and verification in a ZSM environment	Zephyr	Prompt engineering, closed-loop feedback	Intent classification accuracy, processing time on configuration tasks

(Continues)

TABLE 6 | (Continued)

Ref.	Tasks	Proposed	Models	Adaptation techniques	Dataset/ evaluation
Aykurt et al. [52] (2024)	Benchmark of the configuration performance	NetLLMBench, benchmark framework on LLM-based network configuration tasks	LLaMA 3, Mistral, Gemma, Qwen	Prompt engineering, iterative error feedback	Configuration success rate, processing time, and error frequency in the Kathará network emulator
Brodimas et al. [100] (2024)	Intent translation, generate network configurations	LLM-based intent translation and NET JSON generation framework for IBN in 6G networks	GPT-3.5-Turbo	Few-shot learning	Configuration accuracy and deployment time on the Patras5G testbed
Lin et al. [101] (2023)	Intent translation, generate deploy-related code for cloud infrastructure	AppleSeed, transform intents to executable code for network configurations	ChatGPT API (GPT-3 based)	Few-shot learning	Code generation and deployment speed analysis on service chaining and ML pipelining scenarios
Bandara et al. [53] (2024)	Generate configuration (e.g., ACL, network slice related parameters in JSON format)	SliceGPT, network slice management framework using LLM, blockchain, NFT for 5G/6G environment	GPT-3.5-Turbo	Prompt engineering, RAG	Performances of invoke/query transactions, throughput in 5G testbed
Tu et al. [24, 102] (2024,2025)	Intent translation, generate network configuration commands	Intent translation framework to convert intent into low-level configuration commands for IBN	ChatGPT, LLaMA 3.1, Gemma2, Qwen2	LoRA, dynamic in-context learning, continual learning	Accuracy on NFV configuration and formal specification translation use cases
Kou et al. [99] (2025)	Intent abstraction, generate graph-based network intents	GIA, intent abstraction framework using LLM	GPT-3.5, GPT-4, GPT-4o	—	Accuracy to intent abstraction with intent dataset
Ye et al. [104] (2025)	Detect and correct misconfigurations in Kubernetes configurations	LLMSecConfig, LLM and RAG-based context retrieval to generate secure configuration patches	Mistral Large 2, GPT-4o-mini	Prompt engineering, RAG	Correction success rate with 1000 real Kubernetes files from ArtifactHub

processing time and computational resources required when applying the proposed method to real-world networks.

Second, there are LLM-based case studies which use logs of the systems or networks for fault diagnosis and anomaly detection. Qi et al. [107] proposed an anomaly detection framework (LogGPT) which applies preprocessing and prompt engineering (CoT and few-shot learning) of log data based on ChatGPT-3.5-Turbo. The authors evaluated LogGPT with BGL and Spirit datasets,[7] showing better performance of F1-score 0.618 (BGL) and 0.694 (Spirit) compared with baseline models. However, the proposed method has a high false positive rate and hallucination issues, and response time increases as the window size of the log increases.

Zhang et al. [108] presented ScalaLog, which uses GPT-3.5 with CoT and RAG to diagnose types of faults by analyzing historical and real-time logs in an IIoT (Industrial IoT) environment. The authors evaluate ScalaLog compared with baseline studies, performing about 78% F1 score for IoTDB[8] and private datasets. This study proposes the potential of fault diagnosis without log parsing by using LLMs. However, the proposed method shows sensitivity to prompt design due to its reliance on LLMs, and further research is required to optimize processing time and costs for real-time systems. Similarly, Pedroso et al. [109] combined a fine-tuned LLaMA 3 model and dynamic Bayesian networks for log-based anomaly detection and root-cause analysis in a cloud environment.

Last, the other studies suggest the mitigation and recovery methods. Kwon et al. [110] proposed an initial method to correct Ansible script errors using LLM with prompt engineering in an edge cloud environment. The authors evaluated their method with 58 Ansible script error cases (errors with syntax, parameters, logical conditions, etc.), performing 48% accuracy for solving errors. This study shows possibilities for error corrections for Ansible scripts in cloud environment, but still needs further research because of the low correction accuracy. Similarly, Kakarla et al. [56] presented Eywa framework to detect a large number of new faults by analyzing the discrepancy between expected and actual behaviors in DNS protocol tests.

And Hamadanian et al. [111] analyzed network faults and suggested recovery measures in a three-stage structure of hypothesis formation, verification, and mitigation planning through the proposed system. This study presents a theoretical method to apply domain optimization techniques (prompt engineering, fine-tuning, and feedback mechanisms) to LLM in fault management. Also, J. Wang et al. [54] theoretically proposed the NetLM concept for 6G networks, which analyzes network traffic and logs in real time by applying ChatGPT with prompt engineering, RAG, and fine-tuning.

LLM-based approaches in fault management show promising potential for improving fault mitigation performance and automating the fault handling process. These processes include anomaly detection, root cause analysis, and recovery, supported by LLMs' ability to interpret unstructured logs and diverse network data through natural language understanding. Compared with rule-based or ML/DL-based approaches, LLMs offer flexible reasoning and broad contextual awareness across heterogeneous sources. Several studies have demonstrated improvements in diagnostic accuracy by applying techniques such as prompt engineering [107, 109], RAG [108], and feedback mechanisms [106, 111].

Despite these advantages, there still exist some limitations. When trained on insufficient domain-specific data, LLMs can produce high false positive rates. Additionally, the processing time and large model sizes hinder practical use in real-time and edge network environments. In recovery and mitigation scenarios, validation procedures are often not considered yet, which increases the risk of misconfigurations or service disruptions because of the hallucination. Security concerns are also critical. Because LLM-based diagnostic systems operate on sensitive network data, they may pose risks of security threats or data leakages during training or inference, yet many existing studies lack deeper analysis of these issues. Furthermore, most current studies have not been validated in large-scale, dynamic, or multivendor environments, which are limited in the aspect of generalizability.

Future research directions should focus on designing lightweight models which support real-time diagnosis and establishing robust verification pipelines to ensure the safety and correctness of generated outputs. It is also important to build diverse datasets which reflect a variety of fault types and log formats, and to introduce feedback-driven improvement loops.

From a security perspective, it is essential to develop training and inference architectures which incorporate security-preserving mechanisms. Finally, integrating LLMs with reasoning systems or domain-specific engines will be a key challenge to building fault management systems which are both consistent and trustworthy [54, 55]. Table 7 describes the detailed summary of the case studies related to fault management.

4.4 | Security

Network security involves analyzing traffic and logs to detect network attacks and intrusions. Accordingly, most studies focus on analyzing traffic and logs to detect attacks and intrusions. Some studies propose security-specialized language models and tools for generating network intrusion and attack scenarios.

4.4.1 | Traffic and Log Analysis

Traffic and log analysis plays a key role in detecting anomalies and identifying attack patterns. Recent studies have attempted to achieve higher accuracy and automation levels than existing approaches by precisely analyzing network traffic and logs using LLM and adaptation techniques.

First, in traffic analysis using BERT-based models, Lin et al. [112] proposed ET-BERT, which converts network traffic into language tokens and analyzes the relationship between packets and flows through pretraining. Based on ET-BERT, Shi et al. [58] improved the accuracy of service and application classification by extracting packet-level global features using pretrained ET-BERT and learning byte-level local features in parallel using a

TABLE 7 | Summary of case studies for network management: Fault management.

Ref.	Tasks	Proposed	Models	Adaptation techniques	Dataset/evaluation
Kwon et al. [110] (2023)	Fault diagnosis, code correction	Propose a method which uses prompt engineering to improve fault diagnosis and code correction	GPT-3.5, GPT-4, Bard	Prompt engineering	Evaluate with Ansible script data related to fault cases
Hamadanian et al. [111] (2023)	Fault diagnosis, root-cause analysis, mitigation planning	LLM-based iterative reasoning approach for network fault exploration	Not specified	Prompt engineering, Fine-tuning, feedback	Provide theoretical concepts, not evaluated
J. Wang et al. [54] (2023)	Traffic and log analysis, fault diagnosis, generate recovery policy	NetLM, LLM-based multimodal learning to infer fault causes and automate self-healing	ChatGPT-based (not specified)	Prompt engineering, RAG, fine-tuning	Provide theoretical concepts, not evaluated
Kakarla et al. [56] (2023)	Fault diagnosis, protocol model generation	Eywa, LLM-based protocol generation and fault detection framework	GPT-4 with symbolic execution (KLEE)	Prompt engineering	Discovering faults in 10 DNS server implementations
Qi et al. [107] (2023)	Anomaly detection (log-based)	LogGPT, LLM-based anomaly detection methods using system logs	GPT-3.5-Turbo	Few-shot learning, CoT	Accuracies for BGL, Spirit datasets
Shao et al. [55] (2024)	Anomaly detection (frequency pattern), root-cause analysis	WirelessLLM, spectrum anomaly detection and root-cause analysis with LLM	GPT-4, Claude-3 Opus	Few-shot learning	Accuracies for detection in Cognitive Radio environment
Tang et al. [57] (2025)	Fault diagnosis, anomaly detection, generate fault report	MSADM, LLM-based multiscale semanticized anomaly detection in heterogeneous networks	GPT-4, LLaMA 2	CoT	Accuracies for detection and diagnosis with datasets in ns-3

(Continues)

TABLE 7 | (Continued)

Ref.	Tasks	Proposed	Models	Adaptation techniques	Dataset/evaluation
Dharmalingam et al. [106] (2025)	Fault diagnosis, anomaly detection, traffic prediction	Aero-LLM, fault and anomaly detection in UAV networks with 2-tier LLM approach	LLaMA2, OPT, TimesNet, Time-LLM	Fine-tuning (SFT), RLHF	Accuracies for detection and prediction with UAV datasets in real and simulation environment
Zhang et al. [108] (2025)	Log analysis, fault diagnosis	ScalaLog, log–based fault diagnosis framework with summarization and sample augmentation using LLM	GPT-3.5	CoT, RAG	Diagnosis accuracy with Apache IoTDB, Alluxio datasets
Pedroso et al. [109] (2025)	Log analysis, fault diagnosis, anomaly detection	LLM-based log anomaly detection with dynamic Bayesian networks	LLaMA 3	Fine-tuning, RAG-like (cosign similarity)	Accuracies for detection and root-cause analysis in various fault injection scenarios

CNN model. In experiments with the ISCX-VPN dataset[9] and other public datasets, each approach improves service and application classification accuracy (about F1-score 99%) compared with prior methods. However, those studies still have limitations regarding performance degradation possibilities in stronger encryption such as TLS (Transport Layer Security) 1.3 or higher, and need to consider processing time issues in real-time environments. Similarly, Y. Li et al. [113] proposed LLM-Sketch to predict flow sizes and classify large/small flows using the RoBERTa model with LoRA.

Second, in the recent LLM-based traffic analysis such as GPT and LLaMA, Mani et al. [60] presented a method to detect network bottlenecks and anomalies by converting natural language queries into network graph manipulation codes. The authors evaluated the proposed method using their own benchmark tool (NeMoEval), and GPT-4 showed 88% accuracy in NetworkX simulated dataset. However, this study needs more sophisticated prompt engineering or other domain adaptation techniques because the accuracy is reduced as task complexity increases. And there needs to be further research for real-time environment application because this work is evaluated based on a simulation environment.

Kan et al. [63] proposed Mobile-LLaMA for IP routing, packet, and performance analysis in a 5G environment using a LoRA fine-tuned LLaMA 2. The authors used traffic datasets[10] in the 5G environment to analyze and evaluate Mobile-LLaMA with their scoring standards. This study still has limitations regarding hallucination issues in code generation and library usage; the diversity of the dataset; and the protection of data privacy is not considered. Also, the proposed method needs many computational resources for on-premises deployment. Additionally, Marques et al. [86] presented a simple configuration and traffic analysis method to deploy VMs, emulators, and Wireshark using GPT-3.5 with prompt engineering.

Zhou et al. [114] presented a self-refined prompting technique to predict network traffic by having LLM correct prediction errors through its own feedback loop. The authors used the Milan dataset[11] to evaluate the proposed method, performing a 17.09% MAE reduction without pretraining compared with baseline models. However, this study still requires manual processes for complex tasks, has low reproducibility of outputs, and has hallucination issues due to the lack of domain knowledge. As a result, it is necessary to optimize their prompts and apply additional techniques such as fine-tuning and RAG to improve the reasoning of domain knowledge.

In log analysis, Setianto et al. [115] analyzed Cowrie Honeypot[12] log data in real-time using the GPT-2C, which uses GPT-2 small with fine-tuning. The authors evaluated GPT-2C with 29 files of log data from the CyberLab honey dataset,[13] performing an F1-score of 0.89. However, this study does not include consideration for zero-day traffic, and the maximum length of context, such as logs, is limited because they used a small size of LLM.

Wang et al. [116] proposed ShieldGPT to analyze network logs based on the DDoS detection model's outputs and generate security strategies by combining GPT-4 and prompt engineering.

SecureBERT provides attack behavior explanations for detected attack flows, mitigation strategies in natural language, and generates configuration commands. However, LLM of ShieldGPT is an additional component which requires manual implementation for user understanding of the detected attack flows. And ShieldGPT does not have any verification process for the generated commands, so verification is required to check syntax or whether it is applicable and safe in an real-world environment. Similarly, Aghaei et al. [117] designed a security domain-specific language model SecureBERT, pretraining security-related texts using RoBERTa [118] with the custom tokenizer and weight adjustment.

These studies commonly use pretrained language models to effectively extract semantic features from network traffic and log data, thereby implementing high classification accuracy and an automated analysis process. These approaches show promising potential in analyzing network traffic through natural language understanding. However, their deployment in real network environments remains limited due to high computational resource requirements [63, 115] and increased inference latency. These factors hinder their applicability in latency-sensitive scenarios such as real-time analysis or edge computing environments. In addition, most existing studies are conducted on datasets and focus on classification performance [63, 114], while lacking systematic consideration of critical security vulnerabilities of LLMs, such as hallucinations and adversarial inputs which may extract sensitive traffic or log data during analysis. These issues raise concerns regarding the reliability and safety of LLM-based security systems.

Future research should explore the development of lightweight and efficient models for real-world environments, which can reduce processing time and maintain performance [11, 119]. It is also necessary to enhance robustness against adversarial behavior and reduce hallucinations through techniques such as prompt filtering, RAG, and feedback-based validation. Furthermore, improving the explainability of model decision-making can support human supervision and ensure transparency in network security operations. Lastly, strategies such as prompt design for data protection and federated instruction tuning (FedIT) [120] should be considered to overcome current limitations. With these advancements, LLM-based approaches are expected to evolve into secure, reliable, and real-time security solutions for traffic analysis in mission-critical and large-scale networks.

4.4.2 | Attack and Intrusion Detection

Attack and intrusion detection is a key technology that detects malicious behaviors by analyzing abnormal traffic and logs. Most studies detect attacks and intrusions using traditional rule-based approaches or ML/DL-based approaches. Recently, there have been increased attempts to detect attacks and intrusions using LLM with adaptation techniques.

First, several case studies focus on the network attack tools using LLMs. Moskal et al. [121] proposed an attack framework using GPT-3.5-Turbo with CoT to automate network attacks. The authors designed an automated attack loop with a Plan-Act-Report structure and implemented the attack loop in the form of a finite-state machine (FSM). The attack executes attack commands in a Kali Linux environment and collects the results to provide feedback. The proposed method shows a high success rate for well-known vulnerabilities [122] (vsftpd, Samba, etc.), which allows even beginners to implement attack scenarios with the help of LLM and shows the possibility of being useful for automating vulnerability diagnosis. However, this study was evaluated on a simple testbed rather than a real environment and has the limitation of a high failure rate in the exfiltration phase. Similarly, Temara [123] used ChatGPT to automatically collect key information during the reconnaissance phase of penetration testing, thereby increasing efficiency through integration with existing attack tools.

Kholgh and Kostakos [124] presented PAC-GPT, which generates synthetic data for IDS training by automating flow and packet generation using GPT-3. PAC-GPT generates ICMP and DNS Pcap files by using text summaries based on the ToN IoT dataset.[14] In evaluation, the proposed method showed almost 100% accuracy in ICMP scenarios, but the accuracy is lower than 10% in DNS scenarios. The proposed method elevates the possibilities for synthetic data generation for training IDS; however, this study does not include comparisons with existing generative adversarial network (GAN)–based traffic generation methods and still requires more various scenarios such as TCP and UDP. Also, the reproducibility and reliability of generated traffic need to be ensured by establishing synthetic data evaluation standards.

Similarly, Meng et al. [125] proposed a framework as a protocol fuzzing to improve state transition and code coverage, and to suggest the possibility of discovering new security vulnerabilities. Singer et al. [126] used GPT and Gemini to evaluate the possibility of multistage attacks by converting high-level attack commands into accurate low-level commands.

Second, in attack and intrusion detection, BERT/Transformer-based models are often used to detect anomalies and malicious behavior. Case studies compare the proposed methods to traditional rule-based or ML/DL-based approaches. Those studies usually evaluate public datasets to verify their performance.

Yin et al. [127] proposed ExBERT to predict the exploitability based on BERT with transfer learning. ExBERT is trained on datasets describing security vulnerabilities (CVE[15] and ExploitDB[16]) and uses a pooling layer and an LSTM-based classifier to predict the likelihood that a vulnerability will actually be exploited. ExBERT predicts exploitability more accurately (F1-score 91%) than existing SVM and CNN models. This study presents a static analysis approach using vulnerability description documents, so it has limitations in analyzing real-time network traffic or logs. However, ExBERT is notable for understanding semantic context and, compared with previous methods, more accurately reflects the real exploitability of vulnerabilities.

Seyyar et al. [128] detected web attacks using vectorized HTTP requests with BERT and multilayer perceptron (MLP) classifier. The proposed method vectorizes HTTP requests using BERT and inputs the vectorized values into an MLP classifier to detect web attacks such as SQL injection and XSS (cross-site scripting) attacks in real-time. The authors evaluate the proposed method by using the CSIC 2010 dataset,[17] FWAF dataset,[18] and HttpParams dataset;[19] the proposed model showed improved performance (about 98% of F1-score) compared with CNN/RNN-based models. This study cannot analyze encrypted traffic and requires further validation for real-time analysis in deployment, but it contributes to the development of an early language model-based web security system, which can effectively detect network attacks compared with existing web security systems.

Similarly, other studies use BERT and Transformer-based models combined with ML/DL-based models. Diaf et al. [59] combined fine-tuned LLMs (BERT, GPT-2) and LSTM to predict network intrusions in IoT environment. Another study [129] used fine-tuned BART and BERT to predict the possibility of threats by analyzing packet sequences. Manocchio et al. [130] proposed FlowTransformer, a modularized transformer-based detection methods for NIDS. Also, Ferrag et al. [131] presented a lightweight real-time threat detection model with BERT. Adjewa et al. [132] proposed continuous learning-based intrusion detection system combining BERT encoder and Gaussian mixture model (GMM), APT-LLM [133] detected advanced persistent threats (APTs) using LLM and AutoEncoder. Lastly, Hassanina et al. [134] proposed PLLM-CS to detect network traffic anomalies by training long-term context based on Transformer.

Meanwhile, there are other studies to use larger LLMs such as GPT and LLaMA. Zhang et al. [135] introduced the concept of "Generative AI-in-the-loop" and generated synthetic data using GPT-3.5 to train a CNN-based detection. The authors present the roles of LLM in the 4 stages, which consist of the lifecycle of the traditional ML/DL-based models. Also, the authors evaluate the proposed method in the network intrusion detection case study with the DTL-IDS dataset [136], which showed a 28.7% enhanced F1-score compared with the baseline model that does not use LLM. However, the proposed method occasionally has nonsensical or inaccurate outputs, which means hallucination, and self-prompting may affect the performance degradation in such cases. Additionally, it still remains high computational costs and the lack of a validation process for synthetic data.

Ali et al. [137] proposed HuntGPT, combining random forest (RF) classifier and GPT-3.5-Turbo to increase the interpretability of security events. The authors designed an RF-based intrusion detection model and applied XAI techniques (SHAP [138], LIME [139]) to make detection results interpretable. HuntGPT integrates a GPT-3.5-Turbo-based chatbot to explain security events in natural language and suggest response measures. Because HuntGPT was evaluated in the KDD '99 dataset,[20] it lacks coverage of new attack patterns and real-time adaptation. However, an evaluation using Certified Information Security Manager (CISM) exam questions showed a response accuracy of about 72%–82%, demonstrating its potential to support security analysis. Moreover, the XAI-based explanation feature can help improve trust in security detections by making results more interpretable.

Gandhi et al. [140] proposed the SHIELD framework for advanced persistent threat (APT) detection and providing intelligent explanations using LLM. SHIELD uses unsupervised models to detect abnormal events from logs and root-cause localization, and LLM to analyze the sequence and correlation of APT attacks using Qwen2.5 with prompt engineering (CoT). After that, LLM makes a final decision on whether the traffic is an APT attack or not based on its own confidence score. The proposed method compared the performance of LLMs (Qwen2.5, GPT-4o, etc.) with 4 datasets (CADETS,[21] THEIA,[22] Arena,[23] and Blind Eagle[24]), Qwen2.5 showed the best performance (about F1-score 38.9%). This study improved the detection accuracy and explainability of APT attacks using LLM. However, in complex OS environments, the false positive rate can be high, and the configurations of the entire pipeline require many computational resources. In addition, there is a possibility of errors because of the hallucination in the LLM stage, so a sophisticated and consistent prompt design is required.

Similarly, Li et al. [141] detected Carpet Bombing DDoS attacks using semantic analysis of network flows with LLaMA 2, and MSADM [106] analyzed the correlation between network events through an LLM-based intrusion detection with a multiscale attention mechanism. Ghimire et al. [142] presented AutoEncoder-based anomaly detection with GPT-4 preprocessing for improving performance and interpretability. Lastly, Hwang et al. [143] proposed multilabel intrusion detection in Darknet through CG-LLM, which combines TinyLlama with prompt engineering (CoT) and graph neural networks (GNN).

Those case studies highlight a common strength of LLM-based approaches, namely the ability to extract semantic information from traffic and logs to enhance detection accuracy and responsiveness to dynamic and diverse threats. Compared with legacy systems, LLMs can offer flexibility in identifying zero-day or evolving attack patterns. In addition, LLMs have been used not only for detection but also for generating synthetic network attacks and datasets to support IDS training and simulation.

However, several limitations hinder their secure and practical deployment in real-world environments. High computational costs, long inference latency, and hallucination issues can lead to incorrect threat classifications or expose users to unsecure conditions. LLMs are also vulnerable to adversarial prompts and semantic distortions, which may result in false positives or missed attacks [105]. Encrypted traffic presents another blind spot, limiting detection in the networks [58, 128]. Moreover, many studies need to discuss robust security validation mechanisms for accessing and analyzing traffic and log data. Also, only a few provide explainable outputs [137]. These factors indicate that significant improvements in reliability and security are still required for operational integration.

To overcome these limitations, further research directions must integrate security design principles alongside performance optimization. This can include validation processes to verify the correctness of generated outputs or responses. To address hallucinations and adversarial inputs, methods such as feedback-based refinement [42], adversarial training [144, 145], and sophisticated prompt design should be explored. RAG can help reduce hallucination by grounding responses in verified knowledge. Federated learning [120] and encrypted inference techniques should also be considered to protect sensitive network

TABLE 8 | Summary of case studies for network management: Security.

Ref.	Tasks	Proposed	Models	Adaptation techniques	Dataset/evaluation
Yin et al. [127] (2020)	Predict software exploitability and vulnerability	ExBERT, transfer-learning on BERT with pooling+LSTM to predict exploitability	BERT-based	Transfer learning	Accuracies with CVE and ExploitDB datasets
Lin et al. [112] (2022)	Traffic analysis, classification	ET-BERT, encrypted traffic classification model using BERT	BERT (Transformer)-based	Pretraining, Transfer learning	Accuracies with ISCX-VPN and other public datasets
Shi et al. [58] (2023)	Traffic analysis, classification	BFCN, traffic classification method using BERT and CNN	ET-BERT with CNN (not specified)	Pretraining	Accuracies with ISCX-VPN dataset
Mani et al. [60] (2024)	Code conversion, traffic analysis, benchmark	LLM-based automated graph manipulation code and benchmark framework (NemoEval)	GPT-4, GPT-3, text-davinci-003, Bard	Prompt engineering	Accuracies with NeMoEval benchmark
Marques et al. [86] (2024)	Configuration support for traffic analysis	Network analysis and configuration support tool using ChatGPT	GPT-3.5	Prompt engineering	Estimate the number of queries for traffic analysis configurations
Kan et al. [63] (2024)	Traffic analyze for 5G networks	Mobile-LLaMA, LLM-based framework for dynamic network analysis in 5G networks	LLaMA 2	LoRA	Estimate performance in various scenarios (IP routing, packet analysis)
Zhou et al. [114] (2024)	Wireless traffic prediction	LLM-based traffic prediction using feedback loop in wireless networks (traffic prediction part)	GPT-4, GPT-3.5	Prompt engineering, self-refinement	Accuracies with Milan traffic dataset
Y. Li et al. [113] (2025)	Flow sizes prediction, classification	LLM-Sketch, prediction and classification methods for flow sizes with two-tier sketching	RoBERTa	LoRA	Accuracies with CAIDA, MAWI, IMC DC datasets
Setianto et al. [115] (2022)	Real-time log analysis	GPT-2C, real-time log analysis methods based on LLM	GPT-2	Fine-tuning	F1-measure with Cowrie log (SSH attack) datasets

(Continues)

TABLE 8 | (Continued)

Ref.	Tasks	Proposed	Models	Adaptation techniques	Dataset/evaluation
Aghaei et al. [117] (2023)	Data analysis, identify NER for cyber threats	SecureBERT, pretraining RoBERTa with custom tokenizer to analyze security texts	RoBERTa-based	Custom tokenizer, weight adjustment	Accuracy and latency with Cowrie logs dataset (semantic analysis, NER)
Wang et al. [116] (2024)	DDoS detection result analysis, generate mitigation strategies	ShieldGPT, LLM-based attack explanation and mitigation for DDoS attack	GPT-4	Prompt engineering	Efficiency of explanation for detection results, mitigation strategies
Moskal et al. [121] (2023)	Automate multistage network attacks	LLM-based network attack automation methods for scanning and exploitation	GPT-3.5-Turbo	CoT	Success rate of generated attack commands with Kali Linux
Temara [123] (2024)	Automated gathering of reconnaissance information	Automated reconnaissance for penetration testing using ChatGPT	ChatGPT	Prompt engineering	Case studies with reconnaissance requests (SSL/TLS, protocols, etc.)
Kholgh and Kostakos [124] (2023)	Generate synthetic network flows for IDS training	PAC-GPT, synthetic network traffic generation framework for IDS based on LLM	GPT-3	Prompt engineering, fine-tuning	Success rate of ICMP, DNS traffic generation
Meng et al. [125] (2024)	Protocol fuzzing	CHATCFL, enhanced protocol fuzzing methods using LLM	GPT-3.5-Turbo with AFLNET (fuzzer)	Few-shot learning, RAG	Compare performance with AFLNet/NSFuzz
Singer et al. [126] (2025)	Convert high-level abstracted task to low-level attack commands	Incalmo, LLM-based attack execution methods for multistage network attacks	GPT-4o, Sonnet 3.5, Gemini 1.5 Pro	Prompt engineering, iterative execution	Evaluate LLM's multistage attack execution
Seyyar et al. [128] (2022)	Attack and intrusion detection (web attacks)	Web attack detection methods using vectorize HTTP requests with BERT for MLP	BERT with MLP	—	Accuracies with CSIC2010, FWAF, HttpParams datasets
Zhang et al. [135] (2024)	Attack and intrusion detection, synthetic data generation	Generate synthetic malicious traffic for CNN training	GPT-3.5 with CNN	Prompt engineering, Self-evolution prompting	Accuracies with IDS scenarios on OpenStack 5G testbed

(Continues)

TABLE 8 | (Continued)

Ref.	Tasks	Proposed	Models	Adaptation techniques	Dataset/evaluation
Ali et al. [137] (2023)	Explainability of intrusion detection	HuntGPT, Integrate ML-based detection and Explainable AI with LLM	GPT-3.5-Turbo with RF	Prompt engineering	Accuracies with KDD'99, CISM test datasets
Diaf et al. [59] (2024)	Predict intrusions in IoT networks	Intrusion prediction and detection methods for IoT security using DL and LLM	GPT-2, BERT, LSTM	Fine-tuning	Accuracies with CICIoT2023 dataset
Li et al. [141] (2024)	DDoS attack and intrusion detection	DoLLM, LLM-based DDoS attack detection using flow sequences	LLaMA 2	—	Accuracies with CIC-DDoS2019 dataset
Manocchio et al. [130] (2024)	Attack and intrusion detection (NetFlow)	FlowTransformer, Transformer-based NIDS framework	GPT-2, BERT, Shallow 2-layer Transformer	Fine-tuning	Accuracies with NSL-KDD, UNSW-NB15, and CSE-CIC-IDS2018 datasets
Ferrag et al. [131] (2024)	Real-time IoT/IIoT threat detection, lightweight model	SecurityBERT, BERT-based IoT/IIoT threat detection method	BERT with Fixed-Length Encoding	—	Accuracies with Edge-IIoTset dataset
Adjewa et al. [132] (2024)	Attack and intrusion detection	BERT and GMM based continuous intrusion detection methods	BERT with GMM	—	Accuracies with CSE-CIC-IDS2018 dataset
Dharmalingam et al. [106] (2025)	Detection for UAV network security	MSADM, LLM-based multiscale anomaly detection in heterogeneous networks	OPT, LLaMA 2	Fine-tuning, RLHF	Accuracies with TCP/UDP packets and UAV sensor data (MavLink)
Ghimire et al. [142] (2025)	Hybrid anomaly detection with interpretability	Autoencoder-based anomaly detection with GPT-4 preprocessing for IoT environment	GPT-4	Prompt engineering	Accuracies with KDD'99 dataset
Diaf et al. [129] (2025)	Intrusion predictions with packet sequences	LLM-based intrusion prediction framework for IoT networks	BART, BERT	Fine-tuning	Accuracies with CICIoT2023 dataset
Benabder-rahmane et al. [133] (2025)	Detect APT attacks from process events	APT-LLM, embedding-based anomaly detection for APT attacks using LLM	BERT, ALBERT, DistilBERT, RoBERTa, MiniLM	—	Accuracies with DARPA Transparent Computing dataset

(Continues)

TABLE 8 | (Continued)

Ref.	Tasks	Proposed	Models	Adaptation techniques	Dataset/evaluation
Gandhi et al. [140] (2025)	Track and correlate APT attack sequences	SHIELD, LLM-based detection and analysis for APT attacks	Qwen 2.5	CoT	Accuracies with DARPA Eng. 3 CADETS & THEIA, Public Arena, Blind Eagle datasets
Hassanina et al. [134] (2025)	Detect cyber threats in IoT/IIoT networks	PLLM-CS, Transformer-based LLM for long-term threat detection	Transformer-based	—	Accuracies with UNSW_NB15, TON_IoT datasets
Hwang et al. [143] (2025)	Detect darknet threats using multimodal data	CG-LLM, multimodal detection methods for darknet threats using LLM and GNN	TinyLlama with GNN	CoT	Accuracies with CIC-Darknet2020 dataset

data. For encrypted traffic, hybrid models that combine LLMs with traditional protocol analyzers can improve both visibility and accuracy [146]. At last, these security-centric enhancements are essential for ensuring the practical and trustworthy adoption of LLM-based network security systems. Table 8 describes the detailed summary of the case studies related to security.

4.5 | Orchestration

In network orchestration, most of the LLM-based case studies propose automation and optimization methods by translating natural language intents into policies for network management. There are many ways to classify network orchestration tasks. We divide the case studies into two main tasks: (1) intent-based orchestration and automation and (2) resource and performance optimization.

4.5.1 | Intent/Policy-Based Orchestration and Automation

Intent-based orchestration leverages natural language intents of the network administrator into policies using LLM, and operates the network automatically based on the policies. LLM-based case studies propose the methods for intent conversion, intent assurance, lifecycle management, and orchestration of the network functions.

First, several studies focus on natural language intent translation and conversion for network orchestration. J. Wang et al. [54] proposed the theoretical network architecture (NetLM) to convert natural language intents into executable policies based on LLM with prompt engineering, fine-tuning, and RAG. The authors designed NetLM by combining LLMs with ML/DL or RL-based models to represent networks and generate policies for decision-making. Zhang et al. [135] suggested a theoretical concept (Generative AI-in-the-loop) to convert natural language requests into machine-readable policies using LLM in a 5G/B5G environment.

Lin et al. [101] converted intents into executable codes in a multi-cloud environment using the ChatGPT API with few-shot learning. The proposed method uses GPT-3 with few-shot learning to convert the user's natural language intent into Python code, then applies just-in-time (JIT) compilation to generate an execution graph that can run in parallel. The authors evaluate the proposed method with deep packet inspection (DPI) and ML pipeline use cases, performing 2.6 times faster processing time. However, the proposed method needs validation for the accuracy of the generated codes, and the authors conducted the experiments with only 2 intent use cases. Also, the hallucination issue still remains in the intent processing. Similarly, Mani et al. [60] presented a LLM-based management framework to convert natural language requests into executable code and evaluate with the proposed benchmark system (NeMoEval).

Manias et al. [147] combined the LLM (Mistral 7B) and semantic routing method to classify intents and to generate accurate configurations. The authors applied semantic routing to improve LLM-based network automation in intent-based network management and orchestration (IBN-MANO) for 5G core networks. The proposed method showed that the intent classification accuracy was improved by 19%, and the hallucination rate was reduced by 11.5%. This study provides the possibilities of semantic routing for LLM-based intent classification. However, because the semantic routing only supports six predefined static routes, the proposed method needs further research for dynamic or complex intent processing in a real-world environment.

Next, after the intent conversion, some studies have targeted to assure generated policies based on natural language intents, and provide lifecycle management of the intent and policies. Habib et al. [148] proposed a three-stage framework using the ALBERT [149] model, a Transformer prediction model, and Attention-based hierarchical RL (HRL) to avoid intent collisions with network conditions in 5G multi-RAN simulations. In a 5G multi-RAN simulated environment, the proposed method improved throughput by 12%, reduced delay by 26.5%, and improved energy efficiency by 17.1% compared with existing approaches. However, the proposed method needs further validation for the real-time environment and has limitations for context understanding of complex intent because this work uses the lightweight LLM. Also, the hallucination and security issues need to be considered.

Dzeparoska et al. [87] applied the monitor, analyze, plan, execute, knowledge (MAPE-K) loop to automate the LLM-based policy generation and execution process. The authors use LLMs (GPT-3.5 and GPT-4 with few-shot learning) to analyze natural language intents and convert them into executable policies in cloud and network environments. In an OpenStack-based testbed, the experiments showed that the proposed method automatically translated and deployed intents for service chains (DPI, load balancer, web server, etc.), improving execution speed compared with manual orchestration. This study provides the LLM-based policy generation method; however, there needs to be further research for multi-intent conflicts, error detection, and correction for the stability of LLM-generated policies.

Also, their other study [150] detected the differences between the current network state and the target state through KPI-based intent drift modeling. The authors use a GPT-4-based LLM to detect intent drift and generate correct policies. The experiments showed that the proposed method maintained availability at 99.99%, while improving intent drift detection and correction speed by three times. However, this study requires further validation for deployment in real-time environments because of the processing time. In addition, the KPIs are manually defined for each intent, which need standardized and formulated KPIs. Moreover, the generated policies are directly applied to the operational environment without prior testing in a controlled setting such as a digital twin. This raises concerns about potential unintended errors caused by adversarial inputs or hallucinations. Therefore, further research is needed to ensure robust verification of the generated policies.

Similarly, Mekrache and Ksentini [83] presented a method to automate service deployment through NFV orchestrator (NFVO) by converting user intents into NSDs and proposed a human feedback loop to reduce manual intervention. Ayed et al. [79] converted policies into blueprints using GPT-4o and LLaMA 3.1 with CoT and self-refinement. Also, Mekrache et al. [151] decomposed intents by domain and converted intents into infrastructure-level intents using Code LLaMA with prompt engineering and RAG.

Lastly, there exist case studies that discuss the methods to orchestrate the networks using LLMs based on intents and generate policies. GeNet [78] used GPT-4-based automatic configurations when network policies or requirements change, and new devices are added. Chakraborty et al. [51] proposed a theoretical method to deploy network functions to an actual network through NFVO. Hamadanian et al. [111] proposed theoretical methods for predictive maintenance and automatic recovery by monitoring network event logs in real time. Similarly, Dandoush et al. [61] presented a dynamic orchestration method theoretically by converting user intents into network slicing policies. ChatNet [80] conceptually presented dynamic automation of configuration adjustments and resource allocation according to network traffic changes.

These case studies show the applicability of LLM-based intent-to-policy conversion in IBN and automated orchestration. By interpreting high-level natural language requests and translating them into executable orchestration policies or configurations, LLMs can reduce the complexity of network lifecycle management and enhance operational flexibility [147]. Several studies attempt to minimize manual intervention and enable dynamic adaptation to evolving service demands by combining LLMs with existing ML, DL, or RL models [54, 148]. They also incorporate techniques such as feedback loops, semantic routing, and multiagent debate (MAD) to automate tasks including network function or service deployment, SFC [83, 151].

Despite these advantages, most existing studies focus primarily on the conversion of intents into configurations or policies. There need to be more studies on the application of these policies to real-world environments and the implementation of automated network orchestration. In addition, there are several limitations that hinder the operational reliability and scalability of LLM-based orchestration frameworks. First, semantic inconsistency and hallucination issues still remain, especially when processing ambiguous or complex intents. These issues become critical during the conversion from intent to policy or configuration. Many studies do not provide formal validation mechanisms for verifying the correctness of the generated configurations, ensuring compliance with security requirements, or confirming compatibility with legacy infrastructure. This omission may result in performance degradation or operational errors, particularly in multidomain or multivendor environments.

Second, quantitative evaluations on key performance indicators such as execution time, policy application success rate, and system resilience in real-time settings are often missing. The absence of such metrics limits the ability to assess the feasibility of deploying LLM-based orchestration systems in latency-sensitive environments such as 5G/B5G or MEC. Moreover, discussions on integration with existing orchestration frameworks (e.g., ONAP, Nephio, Kubernetes, or OpenStack) are limited, which raises concerns about interoperability and backward compatibility.

Third, there is a lack of comparative analysis on how LLM-based orchestration complements or outperforms traditional rule-based, ML-based, or optimization-based orchestrators. Further studies are needed to identify which orchestration tasks are best addressed by LLMs and under what conditions hybrid approaches are more effective. For example, while LLMs may outperform in configuration scripts generation or service or instance deployment, optimization solvers may still be necessary to achieve resource efficiency and meet service-level agreements during runtime.

To overcome these limitations, future research should focus on establishing policy validation pipelines, integrating LLMs with existing orchestrators and optimization backends, and developing benchmark datasets that reflect realistic orchestration scenarios. For secure orchestration, preventions against adversarial inputs to LLMs must be considered [144]. Lightweight LLMs suitable for edge environments, modular pipeline architectures that separate intent interpretation, policy generation, and execution, and predeployment validation through digital twins can improve both performance and reliability [41]. With these improvements, LLM-based orchestration frameworks can evolve beyond proof-of-concept systems and become trustworthy automation solutions for next-generation network operations.

4.5.2 | Resource and Performance Optimization

Optimizing the network resource and performance plays a key role in efficient network management. Most of the case studies use LLMs with adaptation techniques to understand network administrators' intents and network data and generate policies (maximizing throughput, minimizing latency, load balancing, etc.).

There exist several studies which optimize the resources using LLMs to analyze the network states or select the appropriate model. Wu et al. [152] applied a multimodal encoder which handles various input data using LLaMA 2 with fine-tuning. The proposed framework (NetLLM) uses multimodal encoder models to process various network input data such as text, graphs, and time-series data. NetLLM applies a data-driven low-rank networking adaptation (DD-LRNA) fine-tuning technique based on LLaMA 2 to reduce training costs. The authors evaluate their method on viewport prediction (MAE reduction by 10.1%), adaptive bitrate (ABR) streaming (QoE improvement by 14.5%), and cluster job scheduling (job completion time reduction by 6.8%). However, this study still remains with hallucination issues, and needs to be validated in real-world environments. Also, because the proposed multimodal encoder needs to be manually designed for each task, there are additional costs and domain knowledge for its design and implementation. Therefore, further research is needed to validate whether the proposed approach can fulfill the requirements in real-world environments such as service-level objectives (SLOs), to conduct predeployment verification for hallucination and security risks, and to enhance task generalization such as the usage of multimodal LLMs with pretrained encoders.

FlowMage [62] introduced an LLM-based resource optimization system to solve inefficient core allocation, memory access bottlenecks, and state sharing issues in network function deployment. The proposed method uses LLM to analyze the code of each network function and extract key metadata such as flow definitions, state access intensity, and memory access patterns. This information is then used to configure RSS (Receive-Side Scaling) and determine the parallelization models (e.g., shared-nothing vs. shared-state) to optimize load distribution. The evaluation shows that FlowMage achieves up to 11 times throughput improvement in multicore deployments compared with baseline approaches, by minimizing inter-core state sharing overhead. While this study demonstrates significant throughput improvement by leveraging LLMs for optimization, it also has limitations. The reliance on LLMs may expose the system to hallucination issues, which may lead to inaccurate analysis, performance degradation, or semantic errors. Moreover, the current implementation uses a simple cost function and lacks support for runtime adaptability or dynamic traffic workloads. Future work needs to include fine-tuning LLMs to extract continuous features, integrating deterministic static analysis tools, enabling adaptive RSS configurations, and extending system-level performance metrics such as latency and resource utilization.

Similarly, Du et al. [153] combined the mixture-of-experts (MoE) and LLM (GPT-3.5-Turbo) to analyze user demands and select the optimal DRL-based MoE model. He et al. [44] generated various alternative designs which replace the existing network algorithm through the proposed framework based on GPT-4/3.5 with prompt engineering (CoT); it improves ABR streaming performance compared with baseline methods. Huang et al. [154] attempted to diagnose the cause of performance degradation for AI-based models and optimize the network by fine-tuning GPT-3.5 with CoT for efficient network management in B5G environments.

Other case studies discuss the resource optimization methods applying LLM with existing models. Sun et al. [155] combined GPT-4 and GNN with prompt engineering and a feedback mechanism for optimizing energy consumption in UAV networks. The authors evaluated their method (LLM with GNN) compared with other models (Node2Vec and GAT) [156, 157], performing minimized energy consumption and fast convergence in model training. Although this study demonstrates the potential of LLM-enabled graphs for dynamic network optimization, several limitations remain, including challenges in maintaining real-time data integrity, high computational complexity arising from the integration of LLMs and GNNs, difficulties in aligning heterogeneous representations, and inherent issues such as hallucination in LLM outputs. To address these limitations, future research should explore adaptive network optimization, intelligent policy automation, enhanced threat detection, multimodal integration, lightweight model design, and predeployment verification frameworks.

Tang et al. [158] proposed a combined framework that uses a lightweight LLM and DRL model to support efficient job scheduling in multicloud environments. The authors use the lightweight LLM to generate job candidates to solve the training time and generalization problems of existing RL-based approaches. The proposed method uses Gemma 2 2B, which is fine-tuned with the SPECpower dataset,[25] to generate scheduling action candidates. Then, DQN takes actions considering cost, makespan, and energy consumption. The authors evaluate the proposed method in the CloudSimPlus simulation environment, performing up to a 23% cost reduction, a 15% makespan reduction, and up to a 21.4% energy consumption reduction. However, this study assumes all tasks are independent, whereas real-world environments often involve workflow-based tasks with dependencies. Additionally, the dataset used for LLM fine-tuning is limited to CPU energy consumption, so it needs to consider more various resources such as QoS requirements and latency constraints. Furthermore, this work does not include an analysis of the LLM's accuracy or hallucinations.

Similarly, Noh et al. [159] proposed an adaptive resource optimization framework to generate meta-prompts about channel, queue status, and QoS information of APs using GPT-3.5-Turbo with fine-tuning and closed-loop feedback. Shokrenezhad et al. [160] performed resource orchestration by separating high-level planning and low-level execution by combining RL agents and LLaMA 3.1 with RAG.

Lastly, in terms of optimization of power management, Zou et al. [161] proposed a multiagent generative AI system for power savings through IBN and collective intelligence using on-device LLMs (GPT-4). The authors implement IBN to translate natural language commands into network policies by using collective intelligence among LLM agents. In the energy-saving experiments with a multiagent GPT-4 system in a wireless network, the proposed method achieved over 5% power savings. This study provides the theoretical foundation of a multiagent generative AI framework for designing B5G wireless networks based on collective intelligence. However, further research is needed on lightweight LLMs suitable for on-device environments, designing a cooperative model between agents, training with domain-specific data, implementing secure pipelines between agents, and resolving hallucination issues for reliability.

Zhou et al. [114] proposed power control optimization in wireless networks using LLaMA 3 with adaptation techniques. The authors evaluate the proposed LLM-based approach with iterative prompting compared with the DRL-based approach. In experiments on network power control optimization, the LLaMA-3-based model shows similar average power consumption (about 3.1 W) as the DRL model and faster convergence, thereby reducing the training time. However, this approach still faces several limitations in real-world deployment, including constraints on context window size, high sensitivity to prompt examples, dependency for complex tasks, and potential security vulnerabilities. Furthermore, the success of prompt-based approaches fundamentally relies on the inherent capability of the base LLM, limiting generalizability across domain-specific tasks. Future research needs to explore sophisticated prompt designs to enhance efficiency and security-aware design. Moreover, fine-tuned LLMs with prompt engineering could improve reliability and performance.

Similarly, Lee et al. [162] introduced the concept of "Knowledge-Free Network Management" and provided the power management method with multi-LLM collaboration approaches. WirelessLLM [55] presented optimizing power allocation of OFDM systems using GPT-4 and Claude 3 Opus with prompt engineering, fine-tuning, and RAG. Mongailard et al. [163] proposed a distributed optimization method and converted natural language intents into a power scheduling problem in electric vehicles (EV) charging networks using LLaMA 3.

Case studies on LLM-based resource and performance optimization have demonstrated the potential of LLMs' capabilities for natural language understanding to interpret system requirements and network telemetry for resource allocation. These approaches have shown improvements in throughput, energy efficiency, and cost reduction in simulation environments. However, their deployment in real-world networks remains limited. Most studies rely on simulation environments, and their applicability to large-scale, multidomain, and latency-sensitive settings has not been validated sufficiently.

One of the main challenges in resource and performance optimization is the high computational cost and inference latency of LLMs, which make them difficult to apply in time-sensitive environments such as MEC or B5G networks [55, 114]. In addition, unstable coordination among multiple agents and limited runtime adaptability can degrade performance under dynamic network conditions [162, 163]. When LLMs generate resource policies or scheduling decisions without domain-grounded validation, semantic inconsistencies and hallucinations frequently occur. Yet, most existing studies lack formal validation pipelines or present only minimal mechanisms for ensuring correctness and consistency [159, 160].

Security and privacy also need to be explored. Some LLM-based optimization systems process sensitive telemetry or control data, introducing potential vulnerabilities. In autonomous or closed-loop control settings, adversarial prompts or implicit memorization by LLMs may lead to unintended behaviors or configuration errors [105].

To address these challenges, future research should focus on developing lightweight model architectures and quantized inference strategies suitable for edge and real-time environments [164]. Establishing structured output validation mechanisms and fallback handling strategies will be essential for enhancing the robustness of LLM-generated decision-making. Hybrid frameworks that combine domain-specific techniques such as RAG and fine-tuning can improve inference reliability and offer more explainable and trustworthy solutions for network administrators. Finally, the design of modular, security-aware pipelines which separate generation, validation, and execution will be a key enabler for the secure and reliable adoption of LLM-based optimization in real-world networks [85, 105]. Table 9 describes the detailed summary of the case studies related to orchestration.

5 | Challenges and Future Directions

Despite the benefits of LLM-based network management approaches in interpreting user intents and automating complex network tasks, numerous practical challenges remain unresolved. As previously discussed, LLMs have high inference

TABLE 9 | Summary of case studies for network management: Orchestration.

Ref.	Tasks	Proposed	Models	Adaptation techniques	Dataset/evaluation
Wu et al. [152] (2024)	Viewport prediction, optimized bitrate streaming, job scheduling	NetLLM, LLM-based network management framework using Multi-modal encoder, cost-effective fine-tuning	LLaMA 2	Fine-tuning (DD-LRNA: LoRA-like)	KPIs for Viewport prediction, streaming quality, scheduling
Habib et al. [148] (2024)	Intent analysis and validation, network optimization	LLM-based Intent processing, validation and optimization framework	BERT, Transformer, Hierarchical RL	Prompt engineering	Throughput, delay, energy efficiency on 5G multi-RAN testbed
GeNet [78] (2024)	Automated policy and network configuration	GeNet, LLM-based automated policy configuration	GPT-4	CoT	Accuracies on policy/device configuration scenarios
FlowMage [62] (2024)	Network functions orchestration, state and flow optimization	FlowMage, orchestrate NFs, state and flow optimization framework by dynamic policy adjustment	GPT-3.5, GPT-4, Code LLaMA, Gemini	Prompt engineering	Throughput and latency in NF deployments
Mekrache and Ksentini [83] (2024)	Intent-to-NSD conversion, NFVO deployment automation	LLM-based NSD generation and NFVO automation framework	Code LLaMA	Few-shot Learning, feedback	NFVO-based deployment accuracy, human intervention reduction
Chakraborty et al. [51] (2024)	Automated network orchestration configurations	Automated NFVO deployment, accuracy improvement methods using LLM in 5G environment	GPT-3.5-Turbo, LLaMA 2	Few-shot learning, fine-tuning, input filteration with DNN	Deployment time and configuration accuracy
Mani et al. [60] (2024)	Automated network lifecycle management	LLM-based lifecycle management automation framework, NeMoEval benchmark	GPT-4, GPT-3, text-davinci-003, Bard	Prompt engineering	NeMoEval benchmark, lifecycle management accuracy

(Continues)

TABLE 9 | (Continued)

Ref.	Tasks	Proposed	Models	Adaptation techniques	Dataset/evaluation
Hamadanian et al. [111] (2023)	Integrated fault management and orchestration	OCE-Helper, automated network incident management framework	Not specified	Prompt engineering, Fine-tuning, feedback	Provide theoretical concepts, not evaluated
J. Wang et al. [54] (2023)	Intent-to-policy automation, optimization by network control	NetLM, GPT-4 multimodal SDN policy optimization	ChatGPT-based (not specified)	Prompt engineering, RAG, Fine-tuning	Provide theoretical concepts, not evaluated
Aghaei et al. [117] (2023)	Data analysis, identify NER for cyber threats	SecureBERT, pretraining RoBERTa with custom tokenizer to analyze security texts	RoBERTa-based	Custom tokenizer, weight adjustment	Accuracy and latency with Cowrie logs dataset
Lin et al. [101] (2023)	Intent-to-execution code conversion, multidomain orchestration	AppleSeed, LLM-based JIT orchestration for multicloud	ChatGPT API (GPT-3 based)	Few-shot learning	Deployment time, execution code sizes in multicloud environment
Dzeparoska et al. [87] (2023)	Intent to policy conversion, policy generation	Intent-based application management methods with automated policy generation	GPT-4, GPT-3.5	Few-shot learning, feedback	Automated service chain management by generated policies in OpenStack SAVI testbed
Dzeparoska et al. [150] (2024)	Intent drift detection, automated policy correction in dynamic networks	Intent assurance framework based on intent drift detection using LLM	GPT-4	Few-shot learning, feedback	Automated intent drift detection and intent-management tests in OpenStack SAVI testbed
Manias et al. [147] (2024)	Intent classification, automatic configuration	Semantic routing-based intent classification and optimization	Mistral	—	Intent classification accuracy, processing time with 3GPP standards
Dandoush et al. [61] (2024)	Intent-driven network slicing automation	Multi-agent LLM-based dynamic slicing management	Not specified	Prompt engineering	Provide theoretical concepts, not evaluated

(Continues)

TABLE 9 | (Continued)

Ref.	Tasks	Proposed	Models	Adaptation techniques	Dataset/evaluation
Zhang et al. [135] (2024)	Intent translation, dynamic network policy adjustment	Generative AI-in-the-loop, dynamic intent-based policy execution optimization	Not specified	Prompt engineering	Provide theoretical concepts, not evaluated
Zou et al. [161] (2023)	On-device LLM-driven network automation	Distributed multiagent LLM cooperation for network automation and optimization	GPT-4	Prompt engineering	Evaluate energy savings in wireless network
Ayed et al. [79] (2024)	Blueprint code generation based on policy, network automation	Hermes, policy-based blueprint code generation and automation framework for network digital twins	GPT-4o, LLaMA 3.1	CoT, self-refinement	Blueprint execution accuracy in power and energy-saving scenarios
Zhou et al. [114] (2024)	Wireless network optimization	LLM-based power management in wireless networks	LLaMA 3	Prompt engineering, self-refinement	Power savings and traffic prediction accuracy
Lee et al. [162] (2024)	Collaborative LLM-based optimization	Knowledge-free network operation with multi-LLM collaboration	GPT-3.5-Turbo	In-context learning, CoT	Efficiencies for energy savings
He et al. [44] (2024)	Automatic algorithm design for network optimization	Nada, LLM-driven automated network algorithm design framework	GPT-4, GPT-3.5	CoT	Video straming QoE scenarios with FCC, Starlink, and 4G/5G network trace datasets
Du et al. [153] (2024)	Network optimization (resource and QoS)	Network optimization methods using LLM and MoE	GPT-3.5-Turbo with DRL	Prompt engineering	Resource allocation, QoS optimization scenarios
WirelessLLM [55] (2024)	OFDM power optimization, anomaly detection	LLM-based power allocation optimization in wireless networks	GPT-4, Claude 3 Opus	CoT, few-shot learning, PAL, fine-tuning, RAG	Task scheduling cost, time, and energy benchmarks

(Continues)

TABLE 9 | (Continued)

Ref.	Tasks	Proposed	Models	Adaptation techniques	Dataset/evaluation
Huang et al. [154] (2025)	Network optimization, performance degradation analysis	Automated performance degradation analysis and network optimization in 6G environment	GPT-3.5, T5-based multimodal-CoT architecture	CoT, Zero-shot learning, fine-tuning (student model)	KPIs in 3D rendering scenario under wireless environments, performance degradation analysis
ChatNet [80] (2025)	Dynamic network orchestration	ChatNet, LLM-based dynamic network orchestration framework	Not specified	Prompt engineering, LoRA, RAG	Provide theoretical concepts, not evaluated
Mongaillard et al. [163] (2024)	Resource optimization (power scheduling)	LLM-based user-centric power scheduling optimization method	LLaMA 3, GPT-4	Prompt engineering	Efficiencies for cost and time, intent recognition and classification accuracies with EVRQ data
Sun et al. [155] (2024)	UAV network optimization	UAV network optimization framework using LLM and Graph analysis	GPT-4	Prompt engineering, feedback	UAV path and resource optimization performance in simulation environment
Mekrache et al. [151] (2024)	Intent management	LLM-based Intent lifecycle management architecture for network orchestration	Code LLaMA	Prompt engineering, feedback	Intent translation accuracy, processing time in EURECOM 5G environment
Noh et al. [159] (2025)	Resource optimization	LLM-RAO, adaptive resource optimization framework using LLM in wireless environment	GPT-3.5-Turbo	Prompt engineering, feedback	IEEE 802.11ax uplink performance in simulation
Tang et al. [158] (2025)	Resource optimization	LLM assistant framework for RL-based resource optimization in multicloud environment	Gemma 2 with DQN	Full fine-tuning, LoRA	Cost, makespan, energy consumption in simulation datasets and Google Cluster Trace dataset
Shokrnezhad and Taleb [160] (2025)	Resource optimization, network orchestration	ARC, real-time orchestration and RL-driven QoE optimization in 6G networks	LLaMA 3.1	CoT, few-shot learning, RAG	Resource allocation cost, QoE, system resilience in simulation

latency and computational costs, limiting their use in real-time network operations and edge computing environments. In addition, hallucinations may produce inaccurate outputs and cause security issues. Furthermore, due to difficulties in training on domain-specific network data, it remains impractical to rely exclusively on LLMs to cover all aspects of network management. Therefore, additional research is necessary to address these practical limitations before LLMs can be widely adopted in operational network environments.

In this section, we discuss the facing challenges and open issues related to adaptation in the network management domain of LLM. We present the future direction for AI and LLM applications based on the methodology and case studies.

5.1 | Challenges and Open Issues

5.1.1 | Real-Time Processing and Performances

Although the introduction of LLMs in network management and operations has attracted significant attention, several practical limitations remain. First, real-time processing and performance issues are prominent. Because LLMs demand massive compute cycles, which conflicts with the sub-second control loops expected in carrier-grade networks. In networks that require millisecond-level latency, even the inference latency of LLMs cannot fulfill the SLOs [14, 165]. Although accuracy tends to improve as model size increases, the associated growth in latency and computational cost makes it impractical for resource-constrained edge devices or small network operators to use LLMs.

Memory requirement is also an important issue. Running a 13B parameter model consumes about 26 GB of device memory even before a prompt is loaded. This uses most of the capacity or exceeds the capacity of many switches and customer premises servers, so remote off-loading becomes necessary and the round-trip delay is added. The power consumption of LLMs is also a non-trivial operational cost. There exist attempts to mitigate these challenges through model shrinking techniques such as LoRA [38, 166] and through distilled models [167], yet a significant gap persists between inference delay and the strict real-time requirements of production networks.

For this reason, in 5G/B5G networks with ultra-low latency requirements or in edge IoT environments, an LLM's high inference latency and large memory footprint may become bottlenecks that prevent timely decision-making [165]. Therefore, applying current LLMs to network management without modification faces limitations in terms of scalability and real-time performance, and may cause excessive delays or resource consumption when processing large-scale data and traffic.

5.1.2 | Hallucination and Output Verification

There are many attempts to automate intent translation, network analysis, and management with LLMs; hallucination still remains a fundamental obstacle. For example, misplaced keywords, a non-existent interface name, or an invalid QoS value can flow through a closed-loop controller and propagate errors that trigger service outages. Empirical data from case studies support this concern about hallucination and semantic conflicts [50, 70]. Similar problems appear in log-analysis pipelines, where an incorrect root-cause sentence can steer operators toward the wrong device [55, 115].

Hallucination is particularly acute in networking for the following reasons [90, 168]. First, pretraining corpora contain little consistent configuration grammar or protocol text, so the model extrapolates from fragmentary patterns. Second, the next-token objective rewards linguistic fluency rather than factual fidelity, which lets a command look correct while violating an actual rule. Third, prompts that mingle telemetry, intents, and manuals can exceed the effective attention span and fuse unrelated fragments.

Once the hallucinated output is produced, verification itself becomes a scalability challenge. Confirming syntax, reachability, and idempotence requires extra inference or symbolic verifications [90]. The pipelines which replay every generated change in a digital twin may eliminate most of the hazardous commands [41], but these processes add tens to hundreds of milliseconds of delay per request and consume additional compute cycles. This overhead is tolerable for offline tickets but unsuitable for tasks such as RAN slicing or real-time fault management.

For this reason, there are many efforts that have been attempted to mitigate hallucination. For example, RAG and CoT prompt engineering are among the techniques explored [169]. Nevertheless, most existing studies still report hallucination issues, and the reliability required in production networks has yet to be achieved.

5.1.3 | Domain Specific Adaptation and Legacy System Integration

Domain-specific adaptation and integration with legacy systems present challenges. Current LLMs have been extensively trained on general-domain text, but network operations require specialized data formats such as routing logs, configuration commands, and packet headers, as well as knowledge of communication protocols. LLMs lacking this network-specific knowledge may produce responses that appear linguistically plausible but are inappropriate in actual network contexts [170].

In fact, without a deep understanding of the network domain comparable to a human expert, an LLM can make misleading decisions in error or fault situations. To solve this issue, additional training or fine-tuning on a large volume of network-specific data is required, but such domain adaptation requires enormous data collection efforts and costs [171, 172]. For example, if an LLM trained to perform well in a 5G/B5G network is repurposed for a vehicular network or a cloud network, further training is inevitable due to differences in data formats, and the cost is very high.

In addition, ensuring interoperability with heterogeneous network environments, which include legacy equipment, is also challenging. Appropriate integration of LLM into existing operational support systems (OSS) or SDN/NFV-based platforms requires new frameworks and interfaces. However, it is not easy to secure compatibility between traditional network management tools and the new LLM technology. For this reason, gradual integration with existing systems is essential in the early stages of LLM adaptation, and there are technical challenges to achieving complete automation [173].

5.1.4 | Reliability and Explainablility

While LLMs offer the unique advantage of generating natural language explanations which can explain why a decision and policy has been made, such explanations should be interpreted with caution. Unlike ML/RL-based approaches, which are though less interpretable in natural language, often rely on more structured methods. The textual explanations produced by LLMs are generated based on probabilistic inferences trained from large datasets. So, these explanations may not reflect the true underlying reasoning process and can occasionally cause misleading effects such as hallucination [174, 175]. In network fault management and security event analysis, where accurate interpretation and root-cause analysis are critical, the superficial explainability offered by LLMs does not necessarily translate into trustworthy decision support [176, 177]. Therefore, while the output of LLMs can lead to user understanding, it remains imperative to supplement such explanations with robust validation and verification mechanisms to ensure their reliability.

For example, an LLM-based network operations support system might propose unverified actions for a certain failure scenario or produce responses that overestimate or underestimate a security threat. The challenge is that humans or existing systems would find it difficult to preemptively discern whether such LLM-driven decisions are correct, leading to uncertainty in outcomes. Because of these unexplainable decision processes and potential errors, it is currently recommended to use LLMs as auxiliary tools augmented with human verification, rather than completely replacing human network managers. Unless the transparency and reliability of an LLM's reasoning and decisions can be ensured, there will be inherent limitations in fully deploying LLMs for mission-critical network management.

5.1.5 | Security and Privacy

Security and privacy concerns remain significant challenges in adopting LLMs for network management. Because LLM-based network management systems inherently handle sensitive data such as network configurations, operational logs, user data, and real-time telemetry, these models can become targets of cyber-attacks or unintended sources of sensitive data leaks. In particular, training LLMs on domain-specific datasets can unintentionally lead to the memorization and exposure of confidential information, such as network topology, configuration details, or user access policies [85, 92]. Attackers could exploit vulnerabilities through inference attacks, potentially revealing or reconstructing sensitive data from model outputs [90, 178].

In particular, some studies suggest that sensitive information may remain in model parameters during LLM fine-tuning, which could be exploited by attackers [179, 180]. A critical concern in applying LLMs is adversarial prompt injection, where attackers craft malicious inputs designed to mislead the model into generating hazardous configurations or commands. For example, an attacker might deliberately prompt an LLM-based network configuration to produce configurations that disable security controls or reroute sensitive traffic through compromised paths. Such attacks could lead to service outages, unauthorized access, or data breaches, severely impacting the reliability and security of critical network infrastructures.

To address these issues, existing studies suggest several approaches, including robust prompt filtering mechanisms [145], RAG to verify responses against domain knowledge [181], and rigorous output validation through simulation environments or digital twins [41, 43]. Additionally, methods such as adversarial training can enhance model robustness by explicitly training LLMs to detect and resist malicious inputs. Furthermore, privacy-preserving training approaches such as federated learning and secure multiparty computation (Secure MPC) can minimize the risk of data leaks during model training and inference [105, 182].

When defending against such security threats, ensuring the robustness of LLMs and complying with data usage regulations remain challenges [63]. In particular, when deploying LLMs across distributed edge devices, the varying security levels of each node make it difficult to maintain consistent security controls for the entire system. Handling real-time security responses without compromising model performance and effectively mitigating hallucination-related security vulnerabilities remain open issues. Ultimately, from the perspective of trustworthy AI, careful implementation of privacy protection measures and mitigation of model vulnerabilities are essential for LLM-based network management.

5.2 | Future Directions

As discussed in previous sections, many studies are trying to reduce hallucination and improve the output accuracies of LLMs through domain adaptation techniques in network management and orchestration. Despite these issues, however, future research directions to overcome other challenges and maximize the effectiveness of LLM-based network management include developing lightweight LLMs, improving domain adaptation techniques, integrating with existing technologies, achieving real-time optimization, and enhancing reliability, security, and privacy.

5.2.1 | Efficiency and Lightweight LLMs

There needs further research to make LLM models more lightweight and efficient. This includes developing smaller LLMs capable of running in network edge environments and optimization techniques for faster inference with limited resources. Specifically, methods are being explored to reduce the number of parameters through model compression (e.g., pruning or knowledge distillation) [183, 184], and to increase inference speed without performance degradation by applying quantization to lower numerical precision [164, 185]. In addition, techniques such as continual learning [186] and LoRA [38], which fine-tune only the necessary parts to reduce the retraining cost, and efforts are underway to parallelize model training and reduce costs using distributed learning infrastructure [187]. These lightweight design and optimization techniques are expected to improve the energy efficiency of LLMs, lower the operating cost of large-scale models, and ultimately increase the practicality of deploying them in real network environments.

It is also important to leverage dedicated hardware accelerators such as GPUs and TPUs. Promising directions include introducing specialized inference chipsets or neuromorphic computing technologies [188] to accelerate LLM processing, and incorporating AI acceleration modules into network equipment [189, 190] to ensure real-time performance. For example, embedding a lightweight LLM in a switch or router and optimizing computations in hardware would enable efficient real-time, packet-level analysis and response by the LLM.

5.2.2 | Domain Specific Optimization

Advances in domain adaptation and specialization are required [191]. Network management imposes different requirements depending on the environment, objectives, and functions. Therefore, rather than attempting to handle all network scenarios with a single general-purpose LLM, it may be more effective to adopt a modular LLM architecture or domain-specific training techniques which can be easily adapted to various environments. For example, one could envision an LLM architecture in a mixture-of-experts (MoE) style, composed of modules optimized for each network type (e.g., 5G/6G, vehicular, and edge computing) [173, 192].

In this approach, general understanding of languages would be provided by a general foundation model, while specific modules contribute detailed network expertise and processing; the appropriate LLM module can be invoked depending on the situation. This modular strategy, combined with transfer learning or meta-learning techniques to carry knowledge from one domain to another, aims to enable LLMs to be applied to new network environments with minimal additional training. A recently proposed Configurable LLM framework demonstrates the possibility of supporting multiple specialized domains within a single model by assembling modular LLM components [93].

In addition, practical methodologies for acquiring domain-specific corpora or simulation data to fine-tune LLMs in a customized manner are important [193]. Future research may explore continuous model updates using network log and ticket data (as seen in our case studies) and augmenting LLM training with data generated from virtual network environments. Ultimately, developing efficient domain adaptation techniques that allow LLMs to acquire diverse network domain knowledge with minimal additional cost will be a key challenge in the future.

5.2.3 | Integration With Existing Approaches

Integrating LLMs with existing network management techniques and other AI/ML solutions is a promising direction. Instead of introducing LLMs as completely new standalone systems, an integrated strategy can combine LLMs with currently deployed rule-based systems or machine learning tools to leverage each other's strengths. For example, LLM's natural language understanding can assist the decision-making stage of existing algorithms [101, 194]. A recent study enhanced a traditional LSTM-based intrusion detection system by using an LLM to increase alarm accuracy and provide explainable results [59]. In this way, combining LLMs with established statistical models can exploit the specialized pattern recognition capabilities of those models while utilizing the LLM's reasoning for complex correlation analysis and root-cause diagnosis.

In addition, a collaborative framework which combines LLMs with DRL in network control has been proposed. In this approach, the LLM provides high-level policies or knowledge in natural language, while DRL agents handle low-level control tasks [195]. For example, LLM with IBN can interpret the intents and automatically translate them into network configuration commands; existing SDN controllers or orchestration tools execute those commands. This division of roles helps mitigate the reliability issues observed when using LLMs alone and achieves more stable autonomous network management through cooperation with proven techniques.

Another aspect is the collaboration between human administrators and LLMs. By implementing a human-in-the-loop framework [41, 64] where LLMs automate simple repetitive tasks and large-scale log analysis, and human experts intervene for final approval or exception handling, the risk of LLM errors can be reduced and overall reliability improved. This approach, which combines various intelligent agents including LLMs and humans, is the goal of future network management and is expected to gradually move us closer to fully autonomous networks.

5.2.4 | Distributed Frameworks and Offloading

With model optimization, it is important to leverage edge computing and distributed processing to reduce latency [84, 196]. This is an important factor for improving the real-time response of LLMs and ensuring the QoS of networks and services.

A distributed computing framework is being considered which spreads computation across devices, edge nodes, and cloud servers, rather than relying solely on a central cloud-based LLM. Some studies have proposed collaborative inference techniques that partition the LLM inference process into stages: the first few layers are processed on a local edge node, and the remaining layers are transferred to a nearby server, minimizing latency [197, 198]. This approach can provide a near real-time initial response to data from mobile terminals or sensors, while more complex processing is performed in parallel on the backend, thereby shortening the overall response time. Furthermore, dynamic priority adjustment schemes are being explored, building an asynchronous processing pipeline which immediately handles high-priority network events through a simplified path, while allowing lower-priority tasks to be processed with tolerable delays [199].

For example, if a large-scale network failure is detected, LLM could be designed to provide a preliminary diagnosis in a few seconds, with more detailed analysis following for the emergency response. An architecture that provides LLM services as lightweight microservices through API, allows multiple network management modules to invoke them in parallel, is also being considered to increase overall processing throughput. In short, software optimizations to minimize latency, combined with the adoption of distributed processing infrastructure, should be pursued in parallel. These efforts will broaden the scope of LLM deployment in network management.

5.2.5 | Enhancing Reliability, Security, and Privacy

Finally, enhancing security and privacy is also essential for the sustained advancement of LLM-based network management. Techniques for defending against attacks and protecting data must be explored to reduce vulnerabilities in models. An approach is to train LLMs to be more robust (e.g., adversarial training) so that they produce stable outputs even in the face of malicious inputs or perturbations [144, 145]. For instance, in highly security-sensitive network domains, including data from various simulated attack scenarios in the LLM's training process can help the model build immunity to potential threats.

In addition, integration of explainable AI (XAI) [200] is promised. To provide decision rationales in a human-understandable way, methods such as attention-weight visualization and counterfactual explanations need to be investigated so that administrators can trace the model's reasoning process. Such transparency efforts are particularly useful in network security, as giving human analysts credible explanations for anomalies detected by an LLM can increase trust in the system's responses [201].

Also, privacy-preserving techniques are crucial for LLM deployment. Federated learning can be used to keep sensitive raw data on local devices while sharing only the parameters or gradients [202]. Additionally, approaches such as homomorphic encryption and secure multiparty computation (Secure MPC [203]) enable computation on encrypted data, preventing data exposure during the training process. For example, the lightweight IDS model SecurityBERT [131] achieved high detection accuracy while preserving privacy by transforming IoT device data on-site for training rather than transmitting it to a central server. Applying such privacy-preserving model training and lightweight security frameworks is expected to reduce concerns about personal information leakage when using LLMs and ensure compliance with regulations.

Furthermore, to guarantee the safety of network controls suggested by LLMs, standardized validation procedures need to be established. For instance, important configuration changes proposed by an LLM could be pretested in a sandbox environment, and the model's robustness could be evaluated by injecting certain false inputs into its output stream to observe its behavior. Moreover, issues such as accountability for decisions made by an LLM in network management cannot be ignored. Consequently, guidelines should be developed to clarify responsibilities and to set constraints which LLM-based systems must adhere to. These multifaceted efforts in security and ethics will provide a foundation for LLM-based autonomous networks to be deployed in real-world settings with trust and accountability.

In summary, to successfully apply LLMs in network management and operations, it is necessary to recognize current limitations in real-time performance, scalability, and security, and conduct parallel research efforts to address them. By maximizing the potential of LLMs through lightweight model design, domain adaptation techniques, convergence with existing technologies, and robust security measures, it is expected that an ideal form of autonomous network management can be realized that is reliable with minimizing human interventions. This suggests that next-generation networks such as 6G networks will evolve toward solving complex operational challenges through harmonious collaboration between humans and AI, with LLM technology serving as a central pillar of this transformation.

6 | Conclusion

In this paper, we comprehensively investigated the impact and potential of LLMs on network management. Through case study analysis, we presented that LLMs can provide advanced network management, such as interpreting manager intentions through natural language and automating complex operational tasks, compared with existing AI/ML methods. In particular, the use of LLMs as a solution for network automation has the potential to support high-level decision-making while minimizing human intervention. However, we have also pointed out that LLM-based approaches have practical limitations, such as limitations in real-time processing performance, lack of domain-specific knowledge, integration issues with existing systems, security and privacy risks, and lack of output reliability and explainability.

In order to solve these challenges and maximize the benefits of LLM, we suggest future research directions, such as reducing delays through model weight reduction and improving domain adaptation techniques. With these multifaceted efforts, autonomous network management, in which humans and AI work together to solve complex operational tasks, can be realized in next-generation networks such as 6G. Ultimately, LLM technology is expected to play a key role as a central axis of this transformation, contributing to the establishment of a reliable and responsible network management system.

Acknowledgements

This work was supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (RS-2024-00392332, Development of 6G Network Integrated Intelligence Plane Technologies).

Conflicts of Interest

The authors declare no conflicts of interest.

Data Availability Statement

Data sharing not applicable to this article as no datasets were generated or analysed during the current study.

Endnotes

[1] ONAP: https://www.onap.org/.

[2] OSM: https://osm.etsi.org/.

[3] O-RAN: https://www.o-ran.org/.

[4] GNS3: https://www.gns3.com/.

[5] Kathará: https://www.kathara.org/.

[6] ns-3: https://www.nsnam.org/.

[7] Spirit: https://www.usenix.org/cfdr-data.

[8] IoTDB: https://github.com/AIOps-LogDB/MultiLog-Dataset/.

[9] ISCX-VPN dataset: https://www.unb.ca/cic/datasets/vpn.html.

[10] UE Network Traffic Time-Series (Applications, Throughput, Latency, CQI) in LTE/5G Networks: https://ieee-dataport.org/documents/ue-network-traffic-time-series-applications-throughput-latency-cqi-lte5g-networks.

[11] Milan Dataset: https://ieee-dataport.org/documents/milan-dataset.

[12] Cowrie: https://github.com/cowrie/cowrie.

[13] CyberLab honeynet dataset: https://zenodo.org/records/3687527.

[14] The TON-IoT Datasets: https://research.unsw.edu.au/projects/toniot-datasets.

[15] CVE dataset: https://cve.mitre.org/.

[16] ExploitDB dataset: https://www.exploit-db.com/.

[17] CSIC 2010 dataset: https://www.impactcybertrust.org/dataset_view?idDataset=940.

[18] FWAF dataset: https://www.kaggle.com/datasets/evg3n1j/fwaf-dataset.

[19] HttpParams dataset: https://github.com/enzo-ca/multilabel-url-classification.

[20] KDD Cup 1999 Data: https://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.

[21] DARPA CADETS dataset: https://doi.org/10.7910/DVN/MPUCQU.

[22] DARPA THEIA dataset: https://github.com/darpa-i2o/Transparent-Computing/blob/master/README-E3.md.

[23] Chatbot Arena dataset: https://lmsys.org/blog/2023-07-20-dataset/.

[24] Blind Eagle dataset: https://www.trendmicro.com/en_us/research/21/i/apt-c-36-updates-its-long-term-spam-campaign-against-south-ameri.html.

[25] SPECpower dataset: https://www.spec.org/power_ssj2008/.

References

1. J. Wang, J. Liu, and N. Kato, "Networking and Communications in Autonomous Driving: A Survey," *IEEE Communications Surveys & Tutorials* 21, no. 2 (2019): 1243–1274.

2. R. Mijumbi, J. Serrat, J. L. Gorricho, N. Bouten, F. D. Turck, and R. Boutaba, "Network Function Virtualization: State-of-the-Art and Research Challenges," *IEEE Communication Surveys and Tutorials* 18 (2016): 236–262.

3. N. Feamster, J. Rexford, and E. Zegura, "The Road to SDN: An Intellectual History of Programmable Networks," *ACM SIGCOMM Computer Communication Review* 44, no. 2 (2014): 87–98.

4. B. Han, V. Gopalakrishnan, L. Ji, and S. Lee, "Network Function Virtualization: Challenges and Opportunities for Innovations," *IEEE Communications Magazine* 53, no. 2 (2015): 90–97.

5. S. Lange, N. V. Tu, S. Y. Jeong, et al., "A Network Intelligence Architecture for Efficient VNF Lifecycle Management," *IEEE Transactions on Network and Service Management* 18, no. 2 (2021): 1476–1490.

6. U. J. Umoga, E. O. Sodiya, E. D. Ugwuanyi, et al., "Exploring the Potential of AI-Driven Optimization in Enhancing Network Performance and Efficiency," *Magna Scientia Advanced Research and Reviews* 10, no. 1 (2024): 368–378.

7. R. Boutaba, M. A. Salahuddin, N. Limam, et al., "A Comprehensive Survey on Machine Learning for Networking: Evolution, Applications and Research Opportunities," *Journal of Internet Services and Applications* 9, no. 1 (2018): 1–99.

8. S. Ayoubi, N. Limam, M. A. Salahuddin, et al., "Machine Learning for Cognitive Network Management," *IEEE Communications Magazine* 56, no. 1 (2018): 158–165.

9. A. Leivadeas and M. Falkner, "A Survey on Intent-Based Networking," *IEEE Communications Surveys & Tutorials* 25, no. 1 (2023): 625–655.

10. A. Clemm, L. Ciavaglia, L. Z. Granville, and J. Tantsura, "Intent-Based Networking—Concepts and Definitions," RFC 9315, (2022).

11. W. X. Zhao, K. Zhou, J. Li, et al., "A Survey of Large Language Models," arXiv preprint arXiv:2303.18223 2023.

12. A. Vaswani, N. Shazeer, N. Parmar, et al., "Attention Is All You Need," (2023).

13. P. Lewis, E. Perez, A. Piktus, et al., "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," in Proceedings of the 34th International Conference on Neural Information Processing SystemsNIPS '20. Curran Associates Inc., Red Hook, NY, USA, (2020).

14. T. Mayer, "Future Directions in Cloud Networking for AI and LLM Applications," *Advances in Computer Sciences* 7, no. 1 (2024): 1–7.

15. C. Liu, X. Xie, X. Zhang, and Y. Cui, "Large Language Models for Networking: Workflow, Advances and Challenges," *IEEE Network* 39, no. 5 (2025): 165–172.

16. G. Qu, Q. Chen, W. Wei, Z. Lin, X. Chen, and K. Huang, "Mobile Edge Intelligence for Large Language Models: A Contemporary Survey," (2024).

17. M. Xu, D. Niyato, J. Kang, et al., "Integration of Mixture of Experts and Multimodal Generative AI in Internet of Vehicles: A Survey," (2024).

18. S. Javaid, H. Fahim, B. He, and N. Saeed, "Large Language Models for UAVs: Current State and Pathways to the Future," *IEEE Open Journal of Vehicular Technology* 5 (2024): 1166–1192, https://doi.org/10.1109/OJVT.2024.3446799.

19. S. Long, J. Tan, B. Mao, et al., "A Survey on Intelligent Network Operations and Performance Optimization Based on Large Language Models," *IEEE Communication Surveys and Tutorials* (2025): 1, https://doi.org/10.1109/COMST.2025.3526606.

20. H. Zhou, C. Hu, Y. Yuan, et al., "Large Language Model (LLM) for Telecommunications: A Comprehensive Survey on Principles, Key Techniques, and Opportunities," *IEEE Communication Surveys and Tutorials* 27, no. 3 (2025): 1955–2005.

21. J. Devlin, M. Chang, K. Lee, K. Toutanova, "BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding," CoRR (2018).

22. J. D. Case, M. Fedor, M. L. Schoffstall, and J. Davin, "Simple Network Management Protocol (SNMP)," tech. rep., (1989).

23. R. Hofstede, P. Čeleda, B. Trammell, et al., “Flow Monitoring Explained: From Packet Capture to Data Analysis With Netflow and Ipfix,” *IEEE Communications Surveys & Tutorials* 16, no. 4 (2014): 2037–2064.

24. N. Van Tu, J. H. Yoo, and J. W. K. Hong, “Towards Intent-Based Configuration for Network Function Virtualization Using In-context Learning in Large Language Models,” In: NOMS 2024–2024 IEEE Network Operations and Management Symposium; (2024): 1–8.

25. S. Nambisan and M. Sawhney, “Orchestration Processes in Network-Centric Innovation: Evidence From the Field,” *Academy of Management Perspectives* 25, no. 3 (2011): 40–57.

26. A. Dogra, R. K. Jha, and S. Jain, “A Survey on Beyond 5G Network With the Advent of 6G: Architecture and Emerging Technologies,” *IEEE Access* 9 (2021): 67512–67547.

27. O. Sefraoui, M. Aissaoui, and M. Eleuldj, “OpenStack: Toward an Open-Source Solution for Cloud Computing,” *International Journal of Computer Applications* 55, no. 3 (2012): 38–42.

28. S. Kekki, W. Featherstone, Y. Fang, et al., “MEC in 5G Networks,” *ETSI White Paper* 28, no. 2018 (2018): 1–28.

29. J. Carmigniani and B. Furht, “Augmented Reality: An Overview,” in *Handbook of Augmented Reality*, (Springer New York, 2011), 3–46.

30. J. Hong, S. Park, J. H. Yoo, J. W. K. Hong, “Machine Learning based SLA-Aware VNF Anomaly Detection for Virtual Network Management,” In: 2020 16th International Conference on Network and Service Management (CNSM), (2020): 1–7.

31. D. Y. Lee, S. Y. Jeong, K. C. Ko, J. H. Yoo, and J. W. K. Hong, “Deep Q-Network-Based Auto Scaling for Service in a Multi-Access Edge Computing Environment,” *International Journal of Network Management* 31, no. 6 (2021): e2176.

32. ETSI, “Zero-Touch Network and Service Management (ZSM),” Reference Architecture. Tech. Rep. ETSI GS ZSM 002 V1.1.1, European Telecommunications Standards Institute (ETSI), (2019).

33. 3GPP, “Network Data Analytics Function (NWDAF),” Architecture and Services. Tech. Rep. 3GPP TS 23.288 V18.3.0, 3rd Generation Partnership Project (3GPP), (2023).

34. 3GPP, “Management Data Analytics Function (MDAF) for 5G System,” Tech. Rep. 3GPP TS 28.104 V17.4.0, 3rd Generation Partnership Project (3GPP), (2022).

35. B. K. Saha, D. Tandur, L. Haab, and L. Podleski, “Intent-based Networks: An Industrial Perspective,” in *Proceedings of the 1st International Workshop on Future Industrial Communication NetworksFICN '18*, (Association for Computing Machinery, 2018), 35–40.

36. OpenAI, OpenAI GPT models.

37. Z. Han, C. Gao, J. Liu, J. Zhang, and S. Q. Zhang, “Parameter-Efficient Fine-Tuning for Large Models: A Comprehensive Survey,” arXiv preprint arXiv:2403.14608 2024.

38. E. J. Hu, Y. Shen, P. Wallis, et al., “LoRA: Low-Rank Adaptation of Large Language Models,” (2021). CoRR abs/2106.09685.

39. J. Wei, X. Wang, D. Schuurmans, et al., “Chain-of-Thought Prompting Elicits Reasoning in Large Language Models,” in Advances in Neural Information Processing Systems. 35. Curran Associates, Inc. (2022): 24824–24837.

40. R. Kirk, I. Mediratta, C. Nalmpantis, et al., “Understanding the Effects of RLHF on LLM Generalisation and Diversity,” arXiv preprint arXiv:2310.06452 2023.

41. H. Yang, M. Siew, and C. Joe-Wong, “An LLM-Based Digital Twin for Optimizing Human-in-the Loop Systems,” in *2024 IEEE International Workshop on Foundation Models for Cyber-Physical Systems & Internet of Things (FMSys)*, (IEEE, 2024), 26–31.

42. A. Madaan, N. Tandon, P. Gupta, et al., “Self-Refine: Iterative Refinement With Self-Feedback,” *Advances in Neural Information Processing Systems* 36 (2023): 46534–46594.

43. B. R. Barricelli, E. Casiraghi, and D. Fogli, “A Survey on Digital Twin: Definitions, Characteristics, Applications, and Design Implications,” *IEEE Access* 7 (2019): 167653–167671.

44. Z. He, A. Gottipati, L. Qiu, et al., “Designing Network Algorithms via Large Language Models,” (2024).

45. D. Donadel, F. Marchiori, L. Pajola, and M. Conti, “Can LLMs Understand Computer Networks? Towards a Virtual System Administrator,” (2024).

46. P. Sharma and V. Yegneswaran, “PROSPER: Extracting Protocol Specifications Using Large Language Models,” in *Proceedings of the 22nd ACM Workshop on Hot Topics in Networks*, (Association for Computing Machinery, 2023), 41–47.

47. S. H. G. R. Soman, “Observations on LLMs for Telecom Domain: Capabilities and Limitations,” in *Proceedings of the Third International Conference on AI-ML Systems AIMLSystems '23*, (Association for Computing Machinery, 2024).

48. Holm H, “Bidirectional Encoder Representations From Transformers (BERT) for Question Answering in the Telecom Domain,” Adapting a BERT-like language model to the telecom domain using the ELECTRA pre-training approach. Master's thesis. KTH, School of Electrical Engineering and Computer Science (EECS), (2021).

49. A. Fuad, A. H. Ahmed, M. A. Riegler, and T. Čičić, “An Intent-Based Networks Framework Based on Large Language Models,” in 2024 IEEE 10th International Conference on Network Softwarization (NetSoft), (2024): 7–12.

50. X. Lian, Y. Chen, R. Cheng, et al., “Configuration Validation With Large Language Models,” (2024).

51. S. Chakraborty, N. Chitta, and R. Sundaresan, “Automation of Network Configuration Generation using Large Language Models,” in 2024 20th International Conference on Network and Service Management (CNSM), (2024): 1–7.

52. K. Aykurt, A. Blenk, and W. Kellerer, “NetLLMBench: A Benchmark Framework for Large Language Models in Network Configuration Tasks,” in 2024 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), (2024): 1–6.

53. E. Bandara, P. B. Foytik, S. Shetty, et al., “SliceGPT—OpenAI GPT-3.5 LLM, Blockchain and Non-Fungible Token Enabled Intelligent 5G/6G Network Slice Broker and Marketplace,” 2024 IEEE 21st Consumer Communications & Networking Conference (CCNC), (2024): 439–445.

54. J. Wang, L. Zhang, Y. Yang, et al., “Network Meets ChatGPT: Intent Autonomous Management, Control and Operation,” *Journal of Communications and Information Networks* 8, no. 3 (2023): 239–255.

55. J. Shao, J. Tong, Q. Wu, et al., “WirelessLLM: Empowering Large Language Models Towards Wireless Intelligence,” *Journal of Communications and Information Networks* 9, no. 2 (2024): 99–112.

56. S. K. R. Kakarla and R. Beckett, “Oracle-Based Protocol Testing With Eywa,” (2023).

57. F. Tang, X. Wang, X. Yuan, et al., “Large Language Model (LLM) Assisted End-to-End Network Health Management Based on Multi-Scale Semanticization,” (2025).

58. Z. Shi, N. Luktarhan, Y. Song, and G. Tian, “BFCN: A Novel Classification Method of Encrypted Traffic Based on BERT and CNN,” *Electronics* 12, no. 3 (2023): 516.

59. A. Diaf, A. A. Korba, N. Elislem Karabadji, and Y. Ghamri-Doudane, “Beyond Detection: Leveraging Large Language Models for Cyber Attack Prediction in IoT Networks,” in 2024 20th International Conference on

Distributed Computing in Smart Systems and the Internet of Things (DCOSS-IoT)IEEE Computer Society, (2024): 117–123.

60. S. K. Mani, Y. Zhou, K. Hsieh, S. Segarra, R. Chandra, and S. Kandula, “Enhancing Network Management Using Code Generated by Large Language Models,” (2023).

61. A. Dandoush, V. Kumarskandpriya, M. Uddin, and U. Khalil, “Large Language Models Meet Network Slicing Management and Orchestration,” (2024).

62. H. Ghasemirahni, A. Farshin, M. Scazzariello, M. Chiesa, and D. Kostić, “Deploying Stateful Network Functions Efficiently using Large Language Models,” in *Proceedings of the 4th Workshop on Machine Learning and SystemsEuroMLSys '24*, (Association for Computing Machinery, 2024), 28–38.

63. K. B. Kan, H. Mun, G. Cao, and Y. Lee, “Mobile-LLaMA: Instruction Fine-Tuning Open-Source LLM for Network Analysis in 5G Networks,” *IEEE Network* 38, no. 5 (2024): 76–83.

64. H. Xiao and P. Wang, “LLM a*: Human in the Loop Large Language Models Enabled a* Search for Robotics,” arXiv preprint arXiv:2312.01797 2023.

65. A. Karapantelakis, M. Thakur, A. Nikou, et al., “Using Large Language Models to Understand Telecom Standards,” In: 2024 IEEE International Conference on Machine Learning for Communication and Networking (ICMLCN), (2024): 440–446.

66. H. Zou, Q. Zhao, Y. Tian, et al., “TelecomGPT: A Framework to Build Telecom-Specfic Large Language Models,” (2024).

67. M. Kotaru, “Adapting Foundation Models for Operator Data Analytics,” in *Proceedings of the 22nd ACM Workshop on Hot Topics in Networks*, (Association for Computing Machinery, 2023), 172–179.

68. L. Bariah, H. Zou, Q. Zhao, B. Mouhouche, F. Bader, and M. Debbah, “Understanding Telecom Language Through Large Language Models,” in GLOBECOM 2023–2023 IEEE Global Communications Conference, (2023): 6542–6547.

69. A. L. Bornea, F. Ayed, A. D. Domenico, N. Piovesan, and A. Maatouk, Telco-RAG: Navigating the Challenges of Retrieval-Augmented Language Models for Telecommunications, (2024).

70. C. Wang, M. Scazzariello, A. Farshin, S. Ferlin, D. Kostić, and M. Chiesa, NetConfEval: Can LLMs Facilitate Network Configuration? 2024), Proc. ACM Netw. 2(CoNEXT2).

71. D. M. Manias, A. Chouman, and A. Shami, “Towards Intent-Based Network Management: Large Language Models for Intent Extraction in 5G Core Networks,” (2024).

72. Q. Xiang, Y. Lin, M. Fang, et al., “Toward Reproducing Network Research Results Using Large Language Models,” in *Proceedings of the 22nd ACM Workshop on Hot Topics in NetworksHotNets '23*, (Association for Computing Machinery, 2023), 56–62.

73. L. Zhu, M. Ghobadi, M. Schapira, and S. Shenker, “NCFlow: Reducing Network Congestion in Fat-Tree Data Centers With Distributed Flow Scheduling,” In: Proceedings of the ACM Special Interest Group on Data Communication (SIGCOMM), (2020): 538–551.

74. Y. Zhang, Y. Zhang, H. Yang, X. Yang, H. Yu, and Y. Zhang, “APKeep: Rapid Incremental Updates for Incremental Symbolic Verification of Networks,” *IEEE/ACM Transactions on Networking* 28, no. 3 (2020): 1061–1074.

75. Z. Zhong, M. Ghobadi, A. Khaddaj, J. Leach, Y. Xia, and Y. Zhang, “ARROW: Restoration-Aware Traffic Engineering,” In: Proceedings of the ACM Special Interest Group on Data Communication (SIGCOMM), (2021): 775–788.

76. P. Kazemian, G. Varghese, and N. McKeown, “Header Space Analysis: Unifying Packet Treatment in Computer Networks,” In: Proceedings of the ACM Special Interest Group on Data Communication (SIGCOMM), (2012): 475–486.

77. Z. Wang, J. Araki, Z. Jiang, M. R. Parvez, and G. Neubig, “Learning to Filter Context for Retrieval-Augmented Generation,” (2023).

78. B. Ifland, E. Duani, R. Krief, et al., “GeNet: A Multimodal LLM-Based Co-Pilot for Network Topology and Configuration,” (2024).

79. F. Ayed, A. Maatouk, N. Piovesan, A. D. Domenico, M. Debbah, and Z. Q. Luo, “Hermes: A Large Language Model Framework on the Journey to Autonomous Networks,” (2024).

80. Y. Huang, H. Du, X. Zhang, et al., “Large Language Models for Networking: Applications, Enabling Techniques, and Challenges,” *IEEE Network* 39, no. 1 (2025): 235–242.

81. F. Jiang, Y. Peng, L. Dong, et al., “Large Language Model Enhanced Multi-Agent Systems for 6G Communications,” *IEEE Wireless Communications* 31, no. 6 (2024): 48–55, https://doi.org/10.1109/MWC.016.2300600.

82. H. Quan, W. Ni, T. Zhang, et al., “Large Language Model Agents for Radio Map Generation and Wireless Network Planning,” *IEEE Networking Letters* (2025): 1, https://doi.org/10.1109/LNET.2025.3539829.

83. A. Mekrache and A. Ksentini, “LLM-Enabled Intent-Driven Service Configuration for Next Generation Networks,” In: 2024 IEEE 10th International Conference on Network Softwarization (NetSoft), (2024): 253–257.

84. M. Hosseinzadeh and H. Khamfroush, “DILEMMA: Joint LLM Quantization and Distributed LLM Inference Over Edge Computing Systems,” arXiv preprint arXiv:2503.01704 2025.

85. F. Wu, N. Zhang, S. Jha, P. McDaniel, and C. Xiao, “A New Era in LLM Security: Exploring Security Concerns in Real-World LLM-Based Systems,” arXiv preprint arXiv:2402.18649 2024.

86. S. A. Marques, D. Z. Rodriguez, and R. L. Rosa, “Use of ChatGPT as Configuration Support Tool and Network Analysis,” In: 2023 International Conference on Software, Telecommunications and Computer Networks (SoftCOM), (2023): 1–6.

87. K. Dzeparoska, J. Lin, A. Tizghadam, and A. Leon-Garcia, “LLM-Based Policy Generation for Intent-Based Management of Applications,” In: 2023 19th International Conference on Network and Service Management (CNSM) IEEE, (2023): 1–7.

88. N. Sevim, M. Ibrahim, and S. Ekin, “Large Language Models (LLMs) Assisted Wireless Network Deployment in Urban Settings,” (2024).

89. Y. Wang, M. M. Afzal, Z. Li, et al., “Large Language Model as a Catalyst: A Paradigm Shift in Base Station Siting Optimization,” (2024).

90. L. Huang, W. Yu, W. Ma, et al., “A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions,” *ACM Transactions on Information Systems* 43, no. 2 (2025): 1–55.

91. S. A. Khowaja, P. Khuwaja, K. Dev, H. A. Hamadi, and E. Zeydan, “Pathway to Secure and Trustworthy 6G for LLMs: Attacks, Defense, and Opportunities,” arXiv preprint arXiv:2408.00722 2024.

92. A. Biswas and W. Talukdar, “Guardrails for Trust, Safety, and Ethical Development and Deployment of Large Language Models (LLM),” *Journal of Science and Technology* 4, no. 6 (2023): 55–82.

93. C. Xiao, Z. Zhang, C. Song, et al., “Configurable Foundation Models: Building LLMs From a Modular Perspective,” (2024).

94. C. Wang, M. Scazzariello, A. Farshin, D. Kostic, and M. Chiesa, “Making Network Configuration Human Friendly,” (2023).

95. F. Li, H. Lang, J. Zhang, J. Shen, and X. Wang, “PreConfig: A Pretrained Model for Automating Network Configuration,” (2024).

96. Y. Wei, X. Xie, Y. Zuo, et al., “Leveraging LLM Agents for Translating Network Configurations,” (2025).

97. R. Mondal, A. Tang, R. Beckett, T. Millstein, and G. Varghese, “What Do LLMs Need to Synthesize Correct Router Configurations?” in

Proceedings of the 22nd ACM Workshop on Hot Topics in NetworksHotNets '23, (Association for Computing Machinery, 2023), 189–195.

98. E. D. Jeong, H. G. Kim, S. Nam, J. H. Yoo, and J. W. K. Hong, "S-Witch: Switch Configuration Assistant With LLM and Prompt Engineering," In: NOMS 2024–2024 IEEE Network Operations and Management Symposium, (2024): 1–7.

99. S. Kou, C. Yang, and M. Gurusamy, "GIA: LLM-Enabled Generative Intent Abstraction to Enhance Adaptability for Intent-Driven Networks," *IEEE Transactions on Cognitive Communications and Networking* 11, no. 2 (2025): 999–1012.

100. D. Brodimas, K. Trantzas, B. Agko, et al., "Towards Intent-Based Network Management for the 6G System Adopting Multimodal Generative AI," In: 2024 Joint European Conference on Networks and Communications 6G Summit (EuCNC/6G Summit), (2024): 848–853.

101. J. Lin, K. Dzeparoska, A. Tizghadam, and A. Leon-Garcia, "AppleSeed: Intent-Based Multi-Domain Infrastructure Management via Few-Shot Learning," in 2023 IEEE 9th International Conference on Network Softwarization (NetSoft), (2023): 539–544.

102. N. Tu, S. Nam, and J. W. K. Hong, "Intent-Based Network Configuration Using Large Language Models," *International Journal of Network Management* 35, no. 1 (2025): e2313.

103. O. G. Lira, O. M. Caicedo, and N. L. S. Fonseca, "Large Language Models for Zero Touch Network Configuration Management," *IEEE Communications Magazine* 63 (2024): 146–153.

104. Z. Ye, T. H. M. Le, and M. A. Babar, "LLMSecConfig: An LLM-Based Approach for Fixing Software Container Misconfigurations," (2025).

105. Y. Yao, J. Duan, K. Xu, Y. Cai, Z. Sun, and Y. Zhang, "A Survey on Large Language Model (LLM) Security and Privacy: The Good, the Bad, and the Ugly," *High-Confidence Computing* 4, no. 2 (2024): 100211.

106. B. Dharmalingam, R. Mukherjee, B. Piggott, G. Feng, and A. Liu, "Aero-LLM: A Distributed Framework for Secure UAV Communication and Intelligent Decision-Making," (2025).

107. J. Qi, S. Huang, Z. Luan, et al., "LogGPT: Exploring ChatGPT for Log-Based Anomaly Detection," in *2023 IEEE International Conference on High Performance Computing Communications, Data Science Systems, Smart City Dependability in Sensor, Cloud Big Data Systems Application (HPCC/DSS/SmartCity/DependSys)*, (IEEE Computer Society, 2023), 273–280.

108. L. Zhang, T. Jia, M. Jia, Y. Wu, H. Liu, and Y. Li, "ScalaLog: Scalable Log-Based Failure Diagnosis Using LLM," in ICASSP 2025-2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), (2025): 1–5.

109. D. F. Pedroso, L. Almeida, L. Eduardo, et al., "Anomaly Detection and Root Cause Analysis in Cloud-Native Environments Using Large Language Models and Bayesian Networks," (2025).

110. S. Kwon, S. Lee, T. Kim, D. Ryu, and J. Baik, "Exploring LLM-Based Automated Repairing of Ansible Script in Edge-Cloud Infrastructures," *Journal of Web Engineering* 22, no. 6 (2023): 889–912, https://doi.org/10.13052/jwe1540-9589.2263.

111. P. Hamadanian, B. Arzani, S. Fouladi, et al., "A Holistic View of AI-driven Network Incident Management," in *Proceedings of the 22nd ACM Workshop on Hot Topics in Networks*, (Association for Computing Machinery, 2023), 180–188.

112. X. Lin, G. Xiong, G. Gou, Z. Li, J. Shi, and J. Yu, "ET-BERT: A Contextualized Datagram Representation With Pre-training Transformers for Encrypted Traffic Classification," in *Proceedings of the ACM Web Conference*, (Association for Computing Machinery, 2022), 633–642.

113. Y. Li, Z. Xu, Z. Lv, Y. Hu, Y. Cui, and T. Yang, "LLM-Sketch: Enhancing Network Sketches With LLM," (2025).

114. H. Zhou, C. Hu, D. Yuan, et al., "Large Language Models for Wireless Networks: An Overview From the Prompt Engineering Perspective," (2024).

115. F. Setianto, E. Tsani, F. Sadiq, G. Domalis, D. Tsakalidis, and P. Kostakos, "GPT-2C: A Parser for Honeypot Logs Using Large Pre-Trained Language Models," in *Proceedings of the 2021 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, (Association for Computing Machinery, 2022), 649–653.

116. T. Wang, X. Xie, L. Zhang, C. Wang, L. Zhang, and Y. Cui, "ShieldGPT: An LLM-Based Framework for DDoS Mitigation," in *Proceedings of the 8th Asia-Pacific Workshop on NetworkingAPNet 24*, (Association for Computing Machinery, 2024), 108–114.

117. E. Aghaei, X. Niu, W. Shadid, and E. Al-Shaer, "SecureBERT: A Domain-Specific Language Model for Cybersecurity," in *Security and Privacy in Communication Networks*, (Springer Nature Switzerland, 2023), 39–56.

118. Y. Liu, M. Ott, N. Goyal, et al., "RoBERTa: A Robustly Optimized BERT Pretraining Approach," (2019).

119. J. Kim, H. Lee, G. Ko, et al., "ADOR: A Design Exploration Framework for LLM Serving With Enhanced Latency and Throughput," (2025).

120. J. Zhang, S. Vahidian, M. Kuo, et al., "Towards Building the Federated GPT: Federated Instruction Tuning," (2024).

121. S. Moskal, S. Laney, E. Hemberg, and U. M. O'Reilly, "LLMs Killed the Script Kiddie: How Agents Supported by Large Language Models Change the Landscape of Network Threat Testing," (2023).

122. Ö. Aslan, S. S. Aktuğ, M. Ozkan-Okay, A. A. Yilmaz, and E. Akin, "A Comprehensive Review of Cyber Security Vulnerabilities, Threats, Attacks, and Solutions," *Electronics* 12, no. 6 (2023): 1333.

123. S. Temara, "Maximizing Penetration Testing Success With Effective Reconnaissance Techniques Using ChatGPT," *Asian Journal of Research in Computer Science* 17, no. 5 (2024): 19–29, https://doi.org/10.9734/ajrcos/2024/v17i5435.

124. D. K. Kholgh and P. Kostakos, "PAC-GPT: A Novel Approach to Generating Synthetic Network Traffic With GPT-3," *IEEE Access* 11 (2023): 114936–114951, https://doi.org/10.1109/ACCESS.2023.3325727.

125. R. Meng, M. Mirchev, M. Böhme, and A. Roychoudhury, "Large Language Model guided Protocol Fuzzing," *Network and Distributed System Security Symposium (NDSS)* 2024 (2024): 1–17.

126. B. Singer, K. Lucas, L. Adiga, M. Jain, L. Bauer, and V. Sekar, "On the Feasibility of Using LLMs to Execute Multistage Network Attacks," (2025).

127. J. Yin, M. Tang, J. Cao, and H. Wang, "Apply Transfer Learning to Cybersecurity: Predicting Exploitability of Vulnerabilities by Description," *Knowledge-Based Systems* 210 (2020): 106529.

128. Y. E. Seyyar, A. G. Yavuz, and H. M. Ünver, "An Attack Detection Framework Based on BERT and Deep Learning," *IEEE Access* 10 (2022): 68633–68644, https://doi.org/10.1109/ACCESS.2022.3185748.

129. A. Diaf, A. A. Korba, N. E. Karabadji, Y. Ghamri-Doudane, "BARTPredict: Empowering IoT Security With LLM-Driven Cyber Threat Prediction," (2025).

130. L. D. Manocchio, S. Layeghy, W. W. Lo, G. K. Kulatilleke, M. Sarhan, and M. Portmann, "FlowTransformer: A Transformer Framework for Flow-Based Network Intrusion Detection Systems," *Expert Systems with Applications* 241 (2024): 122564.

131. M. A. Ferrag, M. Ndhlovu, N. Tihanyi, et al., "Revolutionizing Cyber Threat Detection With Large Language Models: A Privacy-Preserving BERT-Based Lightweight Model for IoT/IIoT Devices," *IEEE Access* 12 (2024): 23733–23750.

132. F. Adjewa, M. Esseghir, and L. Merghem-Boulahia, "LLM-Based Continuous Intrusion Detection Framework for Next-Gen Networks," (2024).

133. S. Benabderrahmane, P. Valtchev, J. Cheney, and T. Rahwan. "APT-LLM: Embedding-Based Anomaly Detection of Cyber Advanced Persistent Threats Using Large Language Models," (2025).

134. M. Hassanin, M. Keshk, S. Salim, M. Alsubaie, and D. Sharma, "PLLM-CS: Pre-Trained Large Language Model (LLM) for Cyber Threat Detection in Satellite Networks," *Ad Hoc Networks* 166 (2025): 103645.

135. H. Zhang, A. B. Sediq, A. Afana, and M. Erol-Kantarci, "Generative AI-in-the-Loop: Integrating LLMs and GPTs Into the Next Generation Networks," (2024).

136. B. Farzaneh, N. Shahriar, A. H. Al Muktadir, and M. S. Towhid, "DTL-IDS: Deep Transfer Learning-Based Intrusion Detection System in 5G Networks," in 2023 19th International Conference on Network and Service Management (CNSM), (2023): 1–5.

137. T. Ali and P. Kostakos, "HuntGPT: Integrating Machine Learning-Based Anomaly Detection and Explainable AI With Large Language Models (LLMs)," (2023).

138. K. Roshan and A. Zafar, "Utilizing XAI Technique to Improve Autoencoder Based Model for Computer Network Anomaly Detection With Shapley Additive Explanation (SHAP)," arXiv preprint arXiv:2112.08442 2021.

139. A. M. Salih, Z. Raisi-Estabragh, I. B. Galazzo, et al., "A Perspective on Explainable Artificial Intelligence Methods: SHAP and LIME," *Advanced Intelligent Systems* 7, no. 1 (2025): 2400304.

140. P. A. Gandhi, P. N. Wudali, Y. Amaru, Y. Elovici, and A. Shabtai, "SHIELD: APT Detection and Intelligent Explanation Using LLM," (2025).

141. Q. Li, Y. Zhang, Z. Jia, et al., "DoLLM: How Large Language Models Understanding Network Flow Data to Detect Carpet Bombing DDoS," (2024).

142. A. Ghimire, G. Ghajari, K. Gurung, L. K. Sah, and F. Amsaad, "Enhancing Cybersecurity in Critical Infrastructure With LLM-Assisted Explainable IoT Systems," (2025).

143. Y. Hwang, F. Kurt, F. Curebal, O. Keskin, and A. Subasi, "Contextualgraph-LLM: A Multimodal Framework for Enhanced Darknet Traffic Analysis," Available at SSRN 5099415.

144. S. Xhonneux, A. Sordoni, S. Günnemann, G. Gidel, and L. Schwinn, "Efficient Adversarial Training in LLMs With Continuous Attacks," arXiv preprint arXiv:2405.15589 2024.

145. A. Kumar, C. Agarwal, S. Srinivas, A. J. Li, S. Feizi, and H. Lakkaraju, "Certifying LLM Safety Against Adversarial Prompting," (2025).

146. Y. Ginige, T. Dahanayaka, and S. Seneviratne, "TrafficGPT: An LLM Approach for Open-Set Encrypted Traffic Classification," *Proceedings of the Asian Internet Engineering Conference* 2024 (2024): 26–35.

147. D. M. Manias, A. Chouman, and A. Shami, "Semantic Routing for Enhanced Performance of LLM-Assisted Intent-Based 5G Core Network Management and Orchestration," (2024).

148. M. A. Habib, P. E. I. Rivera, Y. Ozcan, et al., "LLM-Based Intent Processing and Network Optimization Using Attention-Based Hierarchical Reinforcement Learning," (2024).

149. Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "ALBERT: A Lite BERT for Self-Supervised Learning of Language Representations, (2020).

150. K. Dzeparoska, A. Tizghadam, and A. Leon-Garcia, "Intent Assurance Using LLMs Guided by Intent Drift," (2024).

151. A. Mekrache, A. Ksentini, and C. Verikoukis, "Intent-Based Management of Next-Generation Networks: An LLM-Centric Approach," *IEEE Network* 38, no. 5 (2024): 29–36.

152. D. Wu, X. Wang, Y. Qiao, et al., "NetLLM: Adapting Large Language Models for Networking," in *Proceedings of the ACM SIGCOMM 2024 ConferenceACM SIGCOMM 24*, (Association for Computing Machinery, 2024), 661–678.

153. H. Du, G. Liu, Y. Lin, et al., "Mixture of Experts for Network Optimization: A Large Language Model-Enabled Approach," (2024).

154. L. Huang, Y. Wu, D. Simeonidou, "Reasoning AI Performance Degradation in 6G Networks With Large Language Models," (2025).

155. G. Sun, Y. Wang, D. Niyato, et al., "Large Language Model (LLM)-Enabled Graphs in Dynamic Networking," *IEEE Network* 39, no. 4 (2024): 290–301.

156. A. Grover and J. Leskovec, "node2vec: Scalable Feature Learning for Networks," (2016).

157. P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio "Graph Attention Networks," (2018).

158. X. Tang, F. Liu, D. Xu, et al., "LLM-Assisted Reinforcement Learning: Leveraging Lightweight Large Language Model Capabilities for Efficient Task Scheduling in Multi-Cloud Environment," *IEEE Transactions on Consumer Electronics* 71 (2025): 1–5644.

159. H. Noh, B. Shim, and H. J. Yang, "Adaptive Resource Allocation Optimization Using Large Language Models in Dynamic Wireless Environments," (2025).

160. M. Shokrnezhad and T. Taleb, "An Autonomous Network Orchestration Framework Integrating Large Language Models With Continual Reinforcement Learning," (2025).

161. H. Zou, Q. Zhao, L. Bariah, M. Bennis, and M. Debbah, "Wireless Multi-Agent Generative AI: From Connected Intelligence to Collective Intelligence," (2023).

162. H. Lee, M. Kim, S. Baek, N. Lee, M. Debbah, and I. Lee, "Large Language Models for Knowledge-Free Network Management: Feasibility Study and Opportunities," (2024).

163. T. Mongaillard, S. Lasaulce, O. Hicheur, et al., "Large Language Models for Power Scheduling: A User-Centric Approach," (2024).

164. J. Lin, J. Tang, H. Tang, et al., "AWQ: Activation-Aware Weight Quantization for on-Device llm Compression and Acceleration," *Proceedings of Machine Learning and Systems* 6 (2024): 87–100.

165. G. Yang, Q. Ye, and J. Xia, "Unbox the Black-Box for the Medical Explainable AI via Multi-Modal and Multi-Centre Data Fusion: A mini-Review, two Showcases and Beyond," *Information Fusion* 77 (2022): 29–52.

166. T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, "QLoRA: Efficient Finetuning of Quantized LLMs," in *Advances in Neural Information Processing Systems*, vol. 36, eds. A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (Curran Associates, Inc, 2023), 10088–10115.

167. S. T. Sreenivas, S. Muralidharan, R. Joshi, et al., "LLM Pruning and Distillation in Practice: The Minitron Approach," (2024).

168. Z. Zhang, C. Wang, Y. Wang, et al., "LLM Hallucinations in Practical Code Generation: Phenomena, Mechanism, and Mitigation," *Proceedings of the ACM on Software Engineering* 2, no. ISSTA (2025): 481–503.

169. S. Tonmoy, S. Zaman, V. Jain, et al., "A Comprehensive Survey of Hallucination Mitigation Techniques in Large Language Models," arXiv preprint arXiv:2401.01313 2024.

170. C. N. Hang, P. D. Yu, R. Morabito, and C. W. Tan, "Large Language Models Meet Next-Generation Networking Technologies: A Review," *Future Internet* 16, no. 10 (2024): 365.

171. D. Myers, R. Mohawesh, V. I. Chellaboina, et al., "Foundation and Large Language Models: Fundamentals, Challenges, Opportunities, and Social Impacts," *Cluster Computing* 27, no. 1 (2024): 1–26.

172. W. Dai, J. Lin, H. Jin, et al., “Can Large Language Models Provide Feedback to Students? A Case Study on ChatGPT,” In: 2023 IEEE International Conference on Advanced Learning Technologies (ICALT), (2023): 323–325.

173. P. Li, A. Sánchez-Mompó, T. Farnham, A. Khan, and A. Aijaz, “Large Generative AI Models Meet Open Networks for 6G: Integration, Platform, and Monetization,” (2025).

174. Z. Bai, P. Wang, T. Xiao, et al., “Hallucination of Multimodal Large Language Models: A Survey,” (2024).

175. H. Nguyen, Z. He, S. A. Gandre, U. Pasupulety, S. K. Shivakumar, and K. Lerman, “Smoothing Out Hallucinations: Mitigating LLM Hallucination with Smoothed Knowledge Distillation,” arXiv preprint arXiv:2502.11306 2025.

176. H. Wen, P. Sharma, V. Yegneswaran, P. Porras, A. Gehani, and Z. Lin, “6G-XSec: Explainable Edge Security for Emerging OpenRAN Architectures,” In Proceedings of the 23rd ACM Workshop on Hot Topics in Networks, (2024): 77–85.

177. M. Anwar and M. Caesar, “Understanding Misunderstandings: Evaluating LLMs on Networking Questions,” *ACM SIGCOMM Computer Communication Review* 54, no. 4 (2025): 14–24.

178. M. Duan, A. Suri, N. Mireshghallah, et al., “Do Membership Inference Attacks Work on Large Language Models?,” arXiv preprint arXiv:2402.07841 2024.

179. T. Nguyen, H. Nguyen, A. Ijaz, S. Sheikhi, A. V. Vasilakos, and P. Kostakos, “Large Language Models in 6G Security: Challenges and Opportunities,” arXiv preprint arXiv:2403.12239 2024.

180. A. Chaoub and M. Elkotob, “Mobile Network-Specialized Large Language Models for 6G: Architectures, Innovations, Challenges, and Future Trends,” arXiv preprint arXiv:2502.04933 2025.

181. S. Zeng, J. Zhang, P. He, et al., “The Good and the Bad: Exploring Privacy Issues in Retrieval-Augmented Generation (RAG),” arXiv preprint arXiv:2402.16893 2024.

182. P. Xu, H. Xu, M. Chen, Z. Liang, and W. Xu, “Privacy-Preserving Large Language Model in Terms of Secure Computing: A Survey,” in 2025 2nd International Conference on Algorithms, Software Engineering and Network Security (ASENS), (2025): 286–294.

183. W. Wang, W. Chen, Y. Luo, et al., “Model Compression and Efficient Inference for Large Language Models: A Survey,” arXiv preprint arXiv:2402.09748 2024.

184. X. Zhu, J. Li, Y. Liu, C. Ma, and W. Wang, “A Survey on Model Compression for Large Language Models,” *Transactions of the Association for Computational Linguistics* 12 (2024): 1556–1577.

185. Z. Liu, B. Oguz, C. Zhao, et al., “LLM-QAT: Data-Free Quantization Aware Training for Large Language Models,” arXiv preprint arXiv:2305.17888 2023.

186. L. Wang, X. Zhang, H. Su, and J. Zhu, “A Comprehensive Survey of Continual Learning: Theory, Method and Application,” *IEEE Transactions on Pattern Analysis and Machine Intelligence* 46, no. 8 (2024): 5362–5383.

187. F. Brakel, U. Odyurt, A. L. Varbanescu, “Model Parallelism on Distributed Infrastructure: A Literature Review From Theory to LLM Case-Studies,” arXiv preprint arXiv:2403.03699 2024.

188. C. Xiao, Y. Deng, Z. Yang, et al., “LLM-Based Processor Verification: A Case Study for Neuronnorphic Processor,” In: 2024 Design, Automation & Test in Europe Conference & Exhibition (DATE) IEEE, (2024): 1–6.

189. K. Karras, E. Pallis, G. Mastorakis, et al., “A Hardware Acceleration Platform for AI-Based Inference at the Edge,” *Circuits, Systems, and Signal Processing* 39, no. 2 (2020): 1059–1070.

190. P. Shantharama, A. S. Thyagaturu, and M. Reisslein, “Hardware-Accelerated Platforms and Infrastructures for Network Functions: A Survey of Enabling Technologies and Research Studies,” *IEEE Access* 8 (2020): 132021–132085.

191. C. Wang, X. Liu, Y. Yue, et al., “Survey on Factuality in Large Language Models: Knowledge, Retrieval and Domain-Specificity,” arXiv preprint arXiv:2310.07521 2023.

192. N. Patil and K. Malhotra, “Smart Network Management With Cloud-Based AI and Large Language Models,” *Asian American Research Letters Journal* 2, no. 1 (2025): 8–14.

193. M. Alizadeh, M. Kubli, Z. Samei, et al., “Open-Source LLMs for Text Annotation: A Practical Guide for Model Setting and Fine-Tuning,” *Journal of Computational Social Science* 8, no. 1 (2025): 1–25.

194. E. Eigner and T. Händler, “Determinants of LLM-Assisted Decision-Making,” arXiv preprint arXiv:2402.17385 2024.

195. J. Loevenich, E. Adler, T. Hürten, and R. R. F. Lopes, “Design and Evaluation of an Autonomous Cyber Defence Agent Using DRL and an Augmented LLM,” *Computer Networks* 262 (2025): 111162.

196. Y. Shen, J. Shao, X. Zhang, et al., “Large Language Models Empowered Autonomous Edge AI for Connected Intelligence,” *IEEE Communications Magazine* 62, no. 10 (2024): 140–146.

197. C. Liu and J. Zhao, “Resource Allocation in Large Language Model Integrated 6G Vehicular Networks,” In: 2024 IEEE 99th Vehicular Technology Conference (VTC2024-Spring), (2024): 1–6.

198. F. Zhu, F. Huang, Y. Yu, G. Liu, and T. Huang, “Task Offloading With LLM-Enhanced Multi-Agent Reinforcement Learning in UAV-Assisted Edge Computing,” *Sensors* 25, no. 1 (2024): 175.

199. C. Ma, Z. Ye, H. Zhao, et al., “Memory Offloading for Large Language Model Inference With Latency SLO Guarantees,” arXiv preprint arXiv:2502.08182 2025.

200. R. Dwivedi, D. Dave, H. Naik, et al., “Explainable AI (XAI): Core Ideas, Techniques, and Solutions,” *ACM Computing Surveys* 55, no. 9 (2023): 1–33.

201. E. Cambria, L. Malandri, F. Mercorio, N. Nobani, and A. Seveso, “Xai Meets LLMs: A Survey of the Relation Between Explainable AI and Large Language Models,” arXiv preprint arXiv:2407.15248 2024.

202. W. Kuang, B. Qian, Z. Li, et al., “FederatedScope-LLM: A Comprehensive Package for Fine-tuning Large Language Models in Federated Learning,” in *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data MiningKDD '24*, (Association for Computing Machinery, 2024), 5260–5271.

203. D. Rathee, D. Li, I. Stoica, H. Zhang, and R. Popa, “MPC-Minimized Secure LLM Inference,” arXiv preprint arXiv:2408.03561 2024.

Appendix A

Table A1 summarizes the abbreviations that are used in this manuscript.

TABLE A1 | Summary of abbreviations.

Abbreviation	Full name	Abbreviation	Full name
ABR	Adaptive bitrate	AI	Artificial intelligence
AR	Augmented reality	ACL	Access control list
BERT	Bidirectional encoder representations from transformers	BGP	Border gateway protocol
ChatOps	Chat-based operations	CLI	Command line interface
CoT	Chain-of-thought prompting	CNN	Convolutional neural network
DL	Deep learning	DQN	Deep Q-network
DRL	Deep reinforcement learning	EV	Electric vehicle
FSDP	Fully sharded data parallel	FedIT	Federated instruction tuning
GBM	Gradient boosting machine	GPT	Generative pretrained transformer
HRL	Hierarchical reinforcement learning	IBN	Intent-based networking
IDS	Intrusion detection system	IIoT	Industrial Internet of Things
IoT	Internet of Things	IRAG	Intent-based RAG
JIT	Just-in-time	KPI	Key performance indicator
LLaMA	Large language model meta AI	LLM	Large language model
LoRA	Low-rank adaptation	LSTM	Long short-term memory
MAD	Multiagent debate	MDAF	Management data analytics function
MEC	Multiaccess edge computing	ML	Machine learning
MLP	Multilayer perceptron	MoE	Mixture of experts
MSADM	Multiscale semanticized anomaly detection model	NER	Named entity recognition
NF	Network function	NFDAF	Network function data analytics function
NFV	Network function virtualization	NFVO	NFV orchestrator
NSD	Network service descriptor	ONAP	Open network automation platform
OPT	Open pretrained transformer	OSM	Open source MANO
OSPF	Open shortest path first	QA	Question answering
QLoRA	Quantized low-rank adaptation	QoE	Quality of experience
QoS	Quality of service	RAG	Retrieval-augmented generation
RF	Random forest	RL	Reinforcement learning
RLHF	Reinforcement learning from human feedback	RNN	Recurrent neural network
RSS	Receive-side scaling	SDN	Software-defined networking
SFC	Service function chaining	SFT	Supervised fine-tuning
SLO	Service-level objective	SNMP	Simple network management protocol
SQL	Structured query language	UAV	Unmanned aerial vehicle
VM	Virtual machine	VLAN	Virtual local area network
VNF	Virtualized network function	VPP	Verified prompt programming
XAI	Explainable artificial intelligence	XGBoost	Extreme gradient boosting
ZSM	Zero-touch network and service management		

Biographies

Jibum Hong is currently a PhD candidate in the Department of Computer Science and Engineering at Pohang University of Science and Technology (POSTECH), South Korea. He received the B.S. degree from the Department of Computer Science and Engineering, Hanyang University ERICA campus in 2017. He then studied computer science at POSTECH, where he received his Master's degree in 2020. His research interests include applied AI for network management, network monitoring, and fault management.

Nguyen Van Tu received his PhD degree in Computer Science and Engineering at Pohang University of Science and Technology (POSTECH), Korea. He studied electronics and communication at Hanoi University of Science and Technology (HUST), Viet Nam, where he received his BSc degree in 2015. He then studied computer science at POSTECH, Korea, where he received his MSc degree in 2018. His research interests include network and system monitoring, high-performance networks, and applied AI for networking.

James Won-Ki Hong is Professor in the Department of Computer Science and Engineering at Pohang University of Science and Technology (POSTECH). He had worked as the Chief Technology Officer and Senior Executive Vice President for KT (Korea Telecom), the largest telecommunications company in Korea from March 2012 to February 2014, where he was responsible for leading the R&D effort of KT and its subsidiary companies. He was Chairman of National Intelligence Communication Enterprise Association and Chairman of ICT Standardization Committee in Korea. He cofounded and is currently served as Executive Director of SDN/NFV Forum in Korea. His research interests include network innovation, such as software-defined networking and network function virtualization, cloud computing, mobile services, IPTV, ICT convergence technologies (e.g., Smart Home, Smart Energy, and Health care), and Internet of Things. James had served as the Head of Department of Computer Science and Engineering, Dean of Graduate School of Information Technology, Director of POSTECH Information Research Labs, and Head of the Division of IT Convergence Engineering at POSTECH. James received his HBSc and MSc degrees in Computer Science from the University of Western Ontario, Canada, in 1983 and 1985, respectively, and the PhD degree in Computer Science from the University of Waterloo, Canada, in 1991.