GeNet: A Multimodal LLM-Based Co-Pilot for Network Topology and Configuration

Beni Ifland*, Rubin Krief*, Aviram Zilberman*, Elad Duani*, Miro Ohana*, Andres Murillo†, Ofir Manor†, Ortal Lavi†, Kenji Hikichi†, Asaf Shabtai*, Yuval Elovici*, Rami Puzis*
*Ben Gurion University of the Negev, Software and Information Systems Engineering and Cyber@BGU
†Fujitsu

Abstract—**Managing communication networks in enterprise environments is complex, time-consuming, and error-prone. Research on automating network engineering has mainly focused on configuration synthesis, while changes to the physical network topology are often overlooked. In this paper, we introduce *GeNet*, which combines visual and textual information to interpret and modify network topologies and device configurations based on user intents. We evaluated *GeNet* using scenarios adapted from Cisco certification exercises relevant to enterprise networks and a series of different topology image variants. Our results show that *GeNet* can interpret images of network topologies, including low-quality handwritten ones, which can reduce the workload of network engineers and accelerate the network design process. Moreover, *GeNet* demonstrates the ability to handle intents successfully, even with incomplete or varying quality input specifications.**

Index Terms—**Intent-Based Networking, LLM, Co-Pilot, Network Topology, Network Configuration, Multimodal**

I. Introduction

Communication network engineering in enterprise environments is a complex, time-consuming, and error-prone process that relies heavily on large IT, NetOps, and network engineering teams. As modern enterprise network architectures have become more diverse and complex, these tasks have emerged as even more challenging [1]. Researchers from academia and industry have attempted to simplify and automate network engineering workflows. One notable approach is intent-based networking (IBN) [2] which aims to serve as a bridge between high-level network requirements, commonly referred to as intents, and low-level implementations. Originally, IBN frameworks relied on formal languages to express intents and provided tools for optimizing network configurations to satisfy them [3]. Later approaches incorporated natural language processing (NLP)-based methods to reduce the formality, further assisting network engineers [4, 5].

Inspired by advances in the use of co-pilots for programming, such as GitHub Copilot[1] researchers are examining how co-pilots and recommendation systems can be tailored to assist network engineers in network device configuration [6, 7]. However, despite recent advances, much of the research on network engineering automation has mainly focused on configuration synthesis, often overlooking the essential task of network topology design. Engineers testify that whiteboards, though imperfect, are vital for network design[2]. Moreover, Wang et al. [4] suggested that including information such as network topology images, could provide valuable context for generating network configurations. We streamline IBN processes by enabling topology sketches and images to serve directly as network specifications.

In this paper, we introduce *GeNet*, a network engineering co-pilot, and explore how to harness state-of-the-art tools to streamline network engineering workflows in enterprise environments. *GeNet* leverages multimodal large language models (LLMs) to interpret network topology images and device configurations, enabling updates that align with user intents. *GeNet* includes three key modules: the *Topology Understanding* module, the *Intent Implementation* module and an automatic LLM-based evaluation framework, capable of evaluating intent implementation quality with significant correlation to human evaluation.

An example of a typical situation where *GeNet* can assist even novice engineers:

Example 1 (Adding Local PCs): The management has requested the addition of three workstations to a lab, but all ports in the respective switch are occupied. The engineer immediately inputs the above intent, along with an image of the lab's topology and the configurations of its raw components, without the need for special preprocessing. Based on the input, *GeNet* understands that the Ethernet switch in the requested lab has run out of free ports. It suggests adding a switch in a daisy chain topology, connecting the additional workstations to it and outputs the updated configurations and topology.

We conduct a comprehensive assessment of *GeNet*'s ability to handle a variety of intents, by evaluating its performance in a series of scenarios, characterized by varying complexities and input specification quality, formulated based on Cisco certification exercises and interviews with IT experts. These intents include designing and updating network topologies and configurations.

The main highlights of this research are as follows: (1) We present *GeNet*, a multimodal co-pilot designed

[1]https://github.com/features/copilot

[2]https://rule11.tech/the-white-board-and-the-simulation/

2332-5666/25/$31.00 ©2025 IEEE
DOI 10.1109/ICDCSW63273.2025.00026

for enterprise network engineers that facilitates network topology and configuration updates. (2) We developed an automatic LLM-based evaluation framework for IBN intents, which shows a strong and significant correlation with human evaluations. (3) We demonstrate that the capability of LLMs to interpret network topology images is significantly influenced by the quality of the inputs or any missing information. However, regardless of such input issues, *GeNet*'s ability to successfully implement intents isn't significantly affected.

II. Background and Related Work

IBN is a common automation approach that introduces an abstraction layer connecting high-level network requirements with low-level device configurations. IBN emphasizes the desired outcomes of network services rather than describing their implementation [2]. While in traditional IBN systems users have to develop proficiency in using predefined syntax [3, 8], NLP-based approaches offer greater freedom and flexibility [4].

Recent IBN studies examined the use of LLMs to streamline such processes. Furthermore, based on the success of programming assistants such as GitHub Copilot, researchers are developing co-pilots capable of facilitating network engineering tasks. For instance, Zhao et al. [7] presented a framework that generates configurations given natural language intents, utilizing an LLM and retrieval augmented generation (RAG) [9].

Several recent studies have introduced evaluation frameworks to assess LLMs' network reasoning abilities, comparing popular LLMs on networking tasks [4, 10, 11]. Most point out that GPT-4 outperformed other popular models; hence, our research relies on GPT-4. While these studies offer extensive evaluations of LLMs in networking, they have several limitations. Primarily, they focus on simplified and impractical tasks, such as generating small configuration snippets from scratch or translating informal specifications into formal ones. In contrast, we demonstrate the ability to tackle realistic and diverse IBN tasks, which involve modifying existing configurations or topologies comprehensively. Also, we examine realistic networks rather than simple topologies with only a few devices. Moreover, many have utilized JSON-like data structures to convey both initial specifications and final configurations. This approach can simplify processes and improve performance, but it may also be impractical because it requires additional processing of the inputs or the outputs.

While fine-tuning an LLM could enhance its performance in networking tasks [12], it requires specialized datasets that are rare in this domain, as well as frequent retraining to adapt to new technologies and standards. Therefore, in this paper, we explore GPT-4's ability to implement network intents without fine-tuning.

Furthermore, although Donadel et al. [10] reported that LLMs struggle generating topology images, Wang et al. [4] raised the possibility that LLMs' understanding of networks could be enhanced by using topology images or graphs as context. In this research, we aim to explore and build upon these insights and to the best of our knowledge, this is the first attempt to leverage the visual modality by translating network topology images into textual descriptions. Additionally, we are the first to utilize topology paper sketches and evaluate how GPT-4 understands different variants of network topologies.

Finally, contrary to our research, which focuses on comprehensive network topology design and configuration, most related studies have primarily concentrated on generating network configurations. *GeNet* harnesses the capabilities of IBN and multimodal LLMs to deliver user-friendly, holistic assistance to network engineers based on natural language intents and topology images. For a survey on the use of LLMs for networking refer to [13].

III. The *GeNet* Framework

In this section, we describe *GeNet*, a co-pilot designed to assist network engineers in updating network topologies and device configurations. *GeNet* simplifies these tasks by harnessing the capabilities of GPT-4, a state-of-the-art LLM, to provide relevant and effective configuration and topology recommendations, aligned with the user's intents, which we call *GeNet solutions*. *GeNet* receives a natural language intent and network specifications, including the topology image and device configurations, and outputs corresponding solutions.

We designed *GeNet* in a modular fashion, based on the assumption that LLMs demonstrate improved performance on smaller tasks [4]. Figure 1 presents the *GeNet* framework and its key modules: the *Topology Understanding*, the *Intent Implementation*, and its automatic evaluation framework. A description of the modules and our research questions is provided below.

A. *Topology Understanding Module*

In *GeNet*'s first phase, the *topology understanding* module generates a textual representation of the network topology based on the input image. This process uses

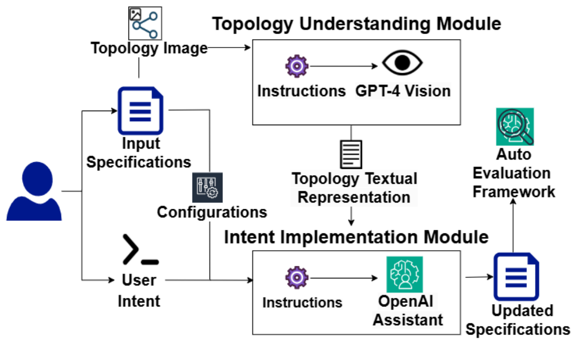

Fig. 1: An overview of the *GeNet* framework

GPT-4's visual question answering (VQA) capabilities[3]. Additionally, the LLM receives a prompt containing precise guidelines, referred to as 'instructions' in fig. 1.

The model is assigned the role of a network topology analyst and instructed to describe the devices and edges in the image, including details such as icons and interface labels. To avoid redundancy, the instructions also specify to exclude unnecessary information, a tendency observed otherwise. These instructions, which serve as the LLM's system prompt, were refined through preliminary experiments to reduce output variability.

B. Intent Implementation Module

In *GeNet*'s second phase, the *intent implementation* module updates the network specifications so that they are aligned with the user's intent. This module utilizes OpenAI's Assistants API[4] which enables the creation of customized AI assistants to perform complex tasks. As before, the assistant is initialized with instructions, referred to as 'instructions' in fig. 1.

The assistant is assigned the role of an expert in network topology and configuration, with precise guidelines regarding its responsibilities. These guidelines include updating the textual representation of the topology to accurately reflect the user's intent, configuring newly added components, and modifying existing device configurations as necessary. Furthermore, the assistant is instructed to explain all changes made during the process to help novice users to build expertise in the domain.

The assistant accesses the textual representation of the topology and the device configurations, performs the necessary updates, and provides the modified files along with corresponding explanations. To maintain the flexibility of *GeNet*, we avoid enforcing a specific configuration language in the instructions. Additionally, GPT-4 is used without fine-tuning to evaluate its inherent capability to effectively implement topology and configuration updates. Notably, *GeNet* is not tailored for GPT-4 and allows for the LLM engine to be easily replaced.

C. Automatic LLM-Based Evaluation

We developed an automatic LLM-based evaluation framework for IBN intent implementations. It is grounded on the hypothesis that it is easier for an LLM to evaluate a *GeNet Solution* using predefined scoring keys than to generate a high-quality solution. Consequently, we expected the LLM to demonstrate strong evaluation capabilities. We implemented this framework using the Assistants API, providing the assistant with instructions about its evaluation task, intent-specific scoring keys (Section IV-C2), and both the input and output network specifications.

For an LLM to function as an effective evaluator, it has to achieve a significant positive correlation with human evaluation. To ensure this, we refined both the instructions and the intent-specific scoring keys in an iterative process (Sec. IV-B). Each time, the LLM was instructed to provide brief explanations for its grades, allowing us to verify its understanding and reasoning process and refine the prompts whenever misunderstandings arose. All instructions and scoring keys mentioned above are available on our Git repository[5].

D. Research Questions

The experiments performed in this work aim to answer the following research questions:

RQ 1: How effectively does GPT-4 interpret network topology images of varying quality? (IV-D2a)

RQ 2: How effectively can a multimodal LLM-based networking co-pilot handle network intents? (IV-D2b)

IV. Experiments

A. Dataset

GNS3 Vault[6] offers training labs featuring scenarios of varying difficulty levels designed to help network engineers prepare for certification exams. Building on this concept, we interviewed IT experts to create a diverse dataset of ten scenarios, each consisting of an intent and network specifications, including device configurations and topology images. Five of them focus on topology updates (adding DMZ, adding DRA, Internet connectivity, adding local PCs, and adding communication servers), while the other five concentrate solely on configuration changes (IP traffic export, basic zone-based firewall, role-based CLI access, time-based access list, and IOS transparent firewall).

To create the network specifications, which provide context for the user intent in each scenario, we simulated the scenarios in a network emulator, extracted the device configurations, and drew the topologies. For a thorough evaluation of GPT-4's ability to interpret network topology images, we created nine topology variants for each scenario. Each topology was created using three visualization formats: 1) GNS3, a network emulator, 2) PowerPoint, a visualization software (without icons for devices), and 3) hand-drawn sketches. Each format was used to create three types of images: 1) *Normal*, 2) *No Labels on Edges*, and 3) *Messy Layout*. The *Normal* variant offers a balanced layout of the topology. The *No Labels on Edges* variant omits edge labels marking device interfaces. The *Messy Layout* variant, which highlights the characteristics of suboptimal visualizations, is created by randomly placing network components on the dashboard or paper and connecting them with edges (see fig. 2). We kept these variants human-readable, avoiding an incomprehensible representation. The complete set of scenarios is publicly available in our Git repository.

[3]https://platform.openai.com/docs/guides/vision
[4]https://platform.openai.com/docs/assistants/overview
[5]https://github.com/networkcopilot
[6]https://gns3vault.com

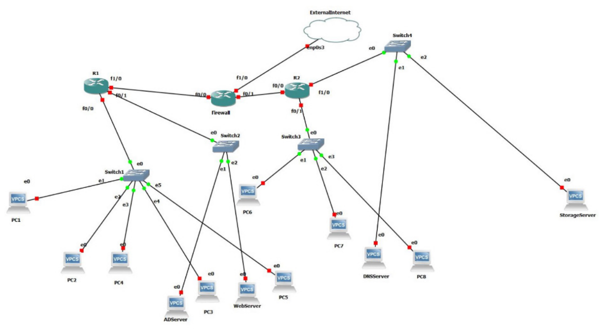

Fig. 2: "Messy Layout" - GNS3 Example

B. Experimental Setup

Every topology image variant in each scenario was processed five times by *GeNet*, resulting in a total of 450 *GeNet solutions* across 90 test cases. Based on preliminary experiments, we employed GPT-4 Turbo with the temperature set to 1 (default value).

While human evaluation is generally considered more reliable, it is also labor-intensive and requires significant effort, which limits the scale of the experiments. To overcome this challenge, we manually evaluated *GeNet solutions* from preliminary experiments using specific scoring criteria (Sec. IV-C2). We used this evaluation to develop and validate our LLM-based evaluation framework (Sec. III-C), which allowed us to scale up our experiments effectively. For this framework, we used the GPT4o model, which was chosen based on preliminary experiments and its cost-effectiveness. A temperature value of 0 was chosen to ensure the evaluation process is deterministic and aligned with the scoring keys.

C. Solution Quality Criteria

1) Topology image understanding score (TIUS): The semi-structured output from the *topology understanding* module was automatically compared to a manually constructed ground-truth representation for each scenario. The TIUS metric quantifies the accuracy with which all visual elements in the topology image were described. To evaluate the LLM's accuracy in identifying elements of the topology, we measured the number of correctly identified elements of a specific type against the total number of elements of that type. The topology elements assessed include: nodes (e.g., routers), referred to as *N*; specific node labels, noted as *NL*; node icons, identified as *NI*; links between nodes, designated as *L*; and the connections of links to nodes (e.g., device interfaces), referred to as *LNL*. The TIUS is calculated as a weighted sum of relative correct assessments:

$$TIUS = 0.3 \cdot N + 0.2 \cdot NL + 0.05 \cdot NI + 0.35 \cdot L + 0.1 \cdot LNL \quad (1)$$

The weights were assigned based on the relative importance of errors in different elements. This score is primarily used to assess the performance of the *Topology Understanding* module.

2) Intent Implementation Score (IIS): Since each scenario includes an intent and network specifications, we developed scenario-specific scoring keys to evaluate how effectively *GeNet* implemented each intent. These scoring keys were initially formulated using the scenario solutions provided by GNS3 Vault for the configuration scenarios and insights from IT expert interviews for the topology scenarios. During the evaluation, additional scoring keys were introduced to account for *GeNet*'s common errors and correct *GeNet solutions* that differed significantly from those provided by GNS3 Vault. When a new valid solution approach or a recurring mistake was identified, the scoring keys were updated, and all prior results were re-evaluated to ensure consistency. Each non-compliance with a key results in a full or partial deduction of the specified points according to the severity of the non-compliance. This scoring system is primarily used to assess *GeNet*'s overall performance.

Example 2 (Adding Local PCs): For the scenario presented as an example (1), the following scoring keys were suggested: 1) Adding a switch to the topology: 20 points. 2) Adding 3 PCs: 10 points per PC. 3) Correct subnet assignment: 3 points. 4) Configuring the new switch: 15 points.

D. Results

During our manual evaluation of *GeNet's Solutions* in the preliminary experiments, we found that out of 250 test cases, *GeNet* successfully configured 79% of the expected network devices. Additionally, *GeNet* reconfigured 92% of the devices from the input specifications that required modifications to comply with intents. Furthermore, it configured 72% of the newly added components that were not included in the initial input. This may suggest that configuring network devices from scratch poses a greater challenge for *GeNet*. In addition, on average, the *Topology Understanding* module processed topology images in 24 seconds each, while the *Intent Implementation* module executed intents in 57 seconds. This results in a total average processing time of 81 seconds per scenario.

1) Automatic LLM-based Evaluation: Various approaches and prompts were tested to optimize the correlation between the LLM and human evaluation. Initially, we instructed the LLM to provide only a final evaluation score of a *GeNet solution* based on the scoring keys, without further explanations, however, this approach resulted in a low correlation. We observed that the LLM often assigned general grades (e.g., 85) to many solutions of varying quality without any rationale. We believe this issue stemmed from the complexity of the evaluation task, which requires comprehension of the grading instructions, scoring keys, and intent of the scenario in the context of both the original and updated

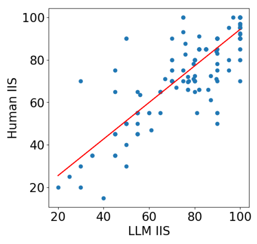

Fig. 3: The correlation between human and LLM IIS

specifications. To improve performance, the evaluation was split into smaller sub-tasks by having the LLM evaluate each scoring key separately. This increased the correlation, though insufficiently. Next, we applied the Chain-of-Though approach [14] by prompting the LLM to provide brief explanations for each scoring key, potentially enhancing its understanding of the task at hand. This led to a significant improvement in alignment with human evaluations, with the Spearman correlation rising to 0.847 ($p = 4.11 \times 10^{-67}$), indicating strong agreement (see fig. 3). We believe that asking the LLM to explain its grading for each key helped it focus on the relevant parts of the *GeNet Solution*, rather than trying to reason about the solution as a whole.

2) *Full experiment:*

a) Topology image understanding (1): Overall, the *topology understanding* module achieved high scores in topology understanding, with a mean TIUS of 0.73 for the *Messy Layout-Paper Sketch* variant, which is considered as the most difficult to comprehend (see fig. 4). This result confirms the potential of multimodal LLM-based co-pilots to effectively utilize network topology images, as raised by [4]. Additionally, as illustrated in fig. 5a and based on a mean equality t-test, the TIUS does not significantly depend on the scenario type.

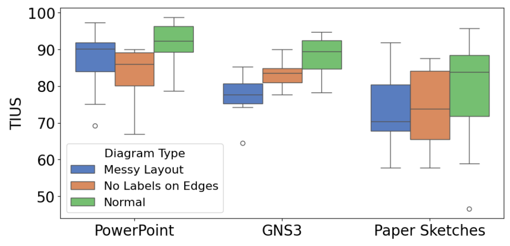

Fig. 4: TIUS by topology format and type

To evaluate the impact of the scenarios, visualization formats, and image types on the TIUS, we measured their information gain (IG). The scenarios alone accounted for most of the variance in the TIUS, with an IG value of 2.25. This can be attributed to the differing topology complexity in each scenario. For instance, the topology in the ”Adding DRA” scenario is considerably more complex than that in the ”Time-Based Access List” scenario. Moreover, as expected, the visualization formats significantly influenced the TIUS. As illustrated in fig. 4, and based on a mean equality t-tests, PowerPoint produced significantly higher TIUS scores compared to images created with GNS3 or sketched on paper. It is not surprising that the *Paper Sketches* yielded the lowest TIUS, as this format lacks the clarity and structure of digital visualizations. However, it is quite surprising that PowerPoint, a general visualization tool, resulted in higher TIUS than GNS3, a specialized networking tool. This might suggest that topologies created with PowerPoint and not containing device icons are more comprehensible for GPT-4. Furthermore, the image types had a notable impact on the TIUS. As anticipated, the *Normal* type resulted in a significantly higher TIUS compared to the *Messy Layout* and *No Labels on Edges* types. However, there was no significant difference in TIUS between the latter two. One possible explanation is that the *Messy Layout* type inherently reduces TIUS due to its suboptimal visualization characteristics, while the *No Labels on Edges* type lacks critical information that could aid in accurately interpreting the topologies. Interestingly, in some instances, the *topology understanding* module was able to perfectly comprehend images of the *Messy Layout* type, achieving a TIUS of 100. This suggests that *GeNet* can interpret low-quality specifications correctly.

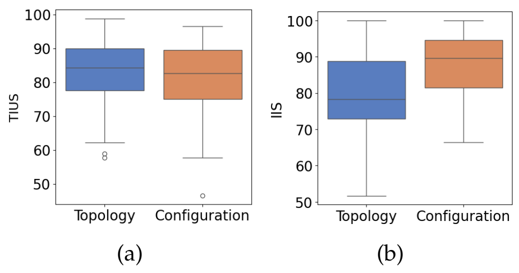

Fig. 5: Scenario type effect on the TIUS (a) and IIS (b)

b) Intent implementation (2): On average, *GeNet* performed better in the configuration scenarios than in the topology scenarios (fig. 5b). A t-test and Levene’s test for mean and variance equality, respectively, indicated significant differences in the IIS between the two scenario types. This suggests that the topology update tasks in our dataset are consistently more challenging for *GeNet* than the configuration ones. One possible explanation is that our configuration scenarios typically involve updating a single device, whereas our topology scenarios require the configuration of multiple devices, hence demand a longer and more complex thought process.

Another potential explanation is that our configuration scenarios do not require devices to be configured from scratch. In contrast, our topology scenarios typically demand adding new devices and configuring them. It is well established that LLMs tend to generate more accurate answers based on a given context or template, as opposed to generating answers from scratch. Thus, configuration scenarios, which mainly involve updating existing device configurations, may be easier for the LLM to handle. In fact, the observations of our preliminary experiment support this assumption (Sec. IV-D).

To deeply investigate the influence of the visualization formats and types on the IIS, we conducted a series of pairwise statistical tests to assess mean and variance equality across different format-type pairs (e.g., GNS3-Messy Layout vs. PowerPoint-Normal). These tests showed insignificant differences in both mean and variance across all pairs regarding the IIS. As illustrated in fig. 6, there is no consistent relationship between these pairs in terms of the IIS. Additionally, the various visualization formats and types exhibited the lowest information gain (IG) concerning the IIS, compared to the TIUS, the scenario, and the scenario type variables. This underscores *GeNet*'s effectiveness in handling network specifications of varying quality (e.g., messy layouts) or with missing information (e.g., interface labels).

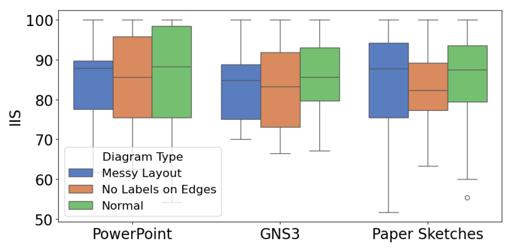

Fig. 6: IIS by topology format and type

V. Conclusion and Future Work

In this paper, we introduce *GeNet*, a multimodal co-pilot designed for enterprise network engineers. *GeNet* is an innovative framework that utilizes an LLM to streamline network design workflows by updating network topologies and device configurations based on user intents. Our results demonstrate that *GeNet* can effectively interpret topology images, highlighting the potential of multimodal LLM-based co-pilots in leveraging such visual data. This can significantly reduce engineers' efforts and accelerate network design processes. Furthermore, *GeNet* shows the ability to successfully handle various intents, even with incomplete or low-quality input specifications, in an average processing time of 81 seconds.

References

[1] Z. B. Houidi and D. Rossi, "Neural language models for network configuration: Opportunities and reality check," *Computer Communications*, vol. 193, pp. 118–125, 2022.

[2] A. Leivadeas and M. Falkner, "A survey on intent based networking," *IEEE Communications Surveys & Tutorials*, 2022.

[3] R. Beckett, R. Mahajan, T. Millstein, J. Padhye, and D. Walker, "Don't mind the gap: Bridging network-wide objectives and device-level configurations," in *Proceedings of the 2016 ACM SIGCOMM Conference*, 2016, pp. 328–341.

[4] C. Wang, M. Scazzariello, A. Farshin, S. Ferlin, D. Kostić, and M. Chiesa, "Netconfeval: Can llms facilitate network configuration?" *Proceedings of the ACM on Networking*, vol. 2, pp. 1–25, 2024.

[5] A. S. Jacobs, R. J. Pfitscher, R. H. Ribeiro, R. A. Ferreira, L. Z. Granville, W. Willinger, and S. G. Rao, "Hey, lumi! using natural language for {intent-based} network management," in *USENIX ATC 21*, 2021, pp. 625–639.

[6] Z. Guo, F. Li, J. Shen, T. Xie, S. Jiang, and X. Wang, "Configreco: Network configuration recommendation with graph neural networks," *IEEE Network*, 2023.

[7] J. Zhao, H. Sun, J. Wang, Q. Qi, Z. Zhuang, S. Tao, and J. Liao, "Confpilot: A pilot for faster configuration by learning from device manuals," in *2023 IEEE 43rd International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2023, pp. 108–119.

[8] S. E. Ooi, R. Beuran, T. Kuroda, T. Kuwahara, R. Hotchi, N. Fujita, and Y. Tan, "Intent-driven secure system design: Methodology and implementation," *Computers & Security*, vol. 124, 2023.

[9] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel *et al.*, "Retrieval-augmented generation for knowledge-intensive nlp tasks," *Advances in Neural Information Processing Systems*, vol. 33, pp. 9459–9474, 2020.

[10] D. Donadel, F. Marchiori, L. Pajola, and M. Conti, "Can llms understand computer networks? towards a virtual system administrator," 2024.

[11] Y. Miao, Y. Bai, L. Chen, D. Li, H. Sun, X. Wang, Z. Luo, D. Sun, and X. Xu, "An empirical study of netops capability of pre-trained large language models," *arXiv preprint arXiv:2309.05557*, 2023.

[12] H. Chen, Y. Miao, L. Chen, H. Sun, H. Xu, L. Liu, G. Zhang, and W. Wang, "Software-defined network assimilation: bridging the last mile towards centralized network configuration management with nassim," in *Proceedings of the ACM SIGCOMM 2022 Conference*, 2022, pp. 281–297.

[13] C.-N. Hang, P.-D. Yu, R. Morabito, and C.-W. Tan, "Large language models meet next-generation networking technologies: A review," *Future Internet*, vol. 16, p. 365, 2024.

[14] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou *et al.*, "Chain-of-thought prompting elicits reasoning in large language models," *Advances in neural information processing systems*, vol. 35, pp. 24 824–24 837, 2022.