On Combining XAI and LLMs for Trustworthy Zero-touch Network and Service Management in 6G

Abdelkader Mekrache, *Member, IEEE,* Mohamed Mekki, *Member, IEEE,* Adlen Ksentini, *Senior Member, IEEE,* Bouziane Brik, *Senior Member, IEEE,* Christos Verikoukis, *Senior Member, IEEE.*

***Abstract*—*Research:* Zero-touch network and Service Management (ZSM) is a key pillar in 6G networks. It allows the 6G management and orchestration framework to operate the networks without external (e.g., human) intervention. To effectively achieve ZSM, advanced network management procedures are required to detect and resolve anomalies within the 6G network autonomously, which usually requires Artificial Intelligence (AI) and Machine Learning (ML) models. However, relying solely on AI can raise concerns about trust due to their lack of explainability. Indeed, as these models are not explainable, it is difficult to understand and trust their decisions. To overcome this limitation, this paper introduces a novel pipeline for ensuring trustworthy ZSM in 6G networks by combining: *(i)* AI for detecting anomalies; *(ii)* eXplainable AI (XAI) to identify the root causes of anomalies using feature importance analysis; and *(iii)* Large Language Models (LLMs) to generate user-friendly explanations and suggest/apply corrective actions to resolve anomalies. A use case is presented using XGBoost as AI, SHAP as XAI, and Llama2 as LLM to address Service Level Agreement (SLA) latency violations within cloud-native 6G microservices. Evaluation results obtained through real experiments demonstrate the framework's efficiency in scaling cloud resources to prevent SLA violations while providing understandable explanations to users, thereby enhancing trust in the system.**

***Abstract*—*Industry:* Zero-touch network and Service Management (ZSM) is key to achieving fully autonomous 6G networks. This paper presents a trusted approach that combines Artificial Intelligence (AI), eXplainable AI (XAI), and Large Language Models (LLMs) to provide ZSM insights and actions. Tested in realistic cloud-native 6G environments, it successfully prevents Service Level Agreement (SLA) violations in microservices applications.**

***Index Terms*—ZSM, 6G, AI, XAI, LLMs, trust, microservices.**

I. Introduction

The emergence of demanding 6G applications like autonomous vehicles and eXtended Reality (XR), along with diverse Quality of Service (QoS) requirements, has significantly increased the complexity of Network Management Systems (NMSs). This complexity emphasizes the necessity of developing the Zero-touch network and Service Management (ZSM) concept, which aims to completely automate and manage future networks without external interventions, such as human interventions. To tackle this challenge, the European Telecommunications Standards Institute (ETSI) established the ZSM group in December 2017. According to their document [1], ZSM-enabled NMSs prioritize the detection and prediction of network anomalies, aiming to resolve them autonomously without human intervention. This approach enables intelligent, autonomous, self-healing networks capable of making informed decisions to resolve anomalies independently. An illustrative example within a typical 6G network involves detecting Service Level Agreement (SLA) violations, such as latency or throughput issues within cloud applications and subsequently identifying the root causes behind these violations. For instance, the system might recognize that insufficient resource allocation (e.g., CPU or RAM) is causing the application's performance to degrade. The solution would then involve autonomously adjusting these resources to mitigate latency or throughput issues, all managed by the NMS without requiring human intervention.

The process of ZSM anomaly resolution involves three steps: *(i)* detecting anomalies, *(ii)* extracting the root cause of anomalies, and *(iii)* resolving the anomalies based on the detected root cause. To address these steps, the research community heavily relies on advanced Artificial Intelligence (AI) methods for anomaly detection [2]. However, ZSM-enabled NMS requires not only detecting but also resolving these anomalies, which necessitates understanding their root causes. This becomes challenging with opaque AI models that lack explainability. For example, using deep learning-based anomaly detection modules provides predictions without insights into how these neural networks arrived at these predictions. To address this limitation, eXplainable AI (XAI) solutions have emerged in research, offering insights into the root causes of anomalies [3]. These models provide information about which features contributed the most to the predictions, thereby revealing the root cause of the anomaly. For instance, if the CPU feature contributed the most to the latency SLA degradation, it indicates insufficient CPU resources for the given application. In this context, *Mekki et al.* in [3] used XAI to explain AI predictions and dynamically scale CPU and RAM resources of a microservice application based on a simple heuristic that leverages XAI values to enable ZSM.

However, XAI often presents explanations in numerical values that may be challenging for users with limited domain knowledge to interpret, impacting the trustworthiness of ZSM systems. User-friendly explanations, on the other hand, aim to generate human-understandable explanations, ensuring they are understandable to diverse users and increasing trust in ZSM's autonomous decisions. Fortunately, with rapid advancements in Generative AI, Large Language Models (LLMs) offer a promising solution. These models can generate human-like

A. Mekrache, M. Mekki, and A.Ksentini are with EURECOM, France (e-mail: abdelkader.mekrache@eurecom.fr, mohamed.mekki@eurecom.fr, adlen.ksentini@eurecom.fr).

B. Brik is with the Computer Science Department, College of Computing and Informatics, Sharjah University, UAE (e-mail: bbrik@sharjah.ac.ae).

C. Verikoukis is with ISI/ATH, University of Patras, Greece (e-mail: chverik@gmail.com).

text in various human languages [4], making them an attractive choice. Moreover, LLMs excel in reasoning tasks without requiring additional training [4], which can be leveraged to resolve anomalies, enabling trustworthy ZSM autonomously. Thus, LLMs should be explored for decision-making problems in 6G networks (e.g., resolving anomalies).

To address the aforementioned challenges, we extend the work in [3] by proposing a trustworthy ZSM-enabled NMS design based on a novel anomaly detection and resolution pipeline: AI|XAI|LLM. Furthermore, we present a pipeline's use case wherein the NMS dynamically scales cloud resources (e.g., CPU and RAM) for a microservice application with specific SLA latency requirements. To achieve this, we employed: *(i)* XGBoost [5] as the AI model to predict SLA latency violations; *(ii)* SHAP [6] as the XAI model to provide numerical explanations of these anomalies, revealing their root causes; and *(iii)* Llama2 [7] as the LLM to generate human-friendly explanations using natural language. Subsequently, we consider two scenarios: if ZSM is enabled, Llama2 autonomously resolves the anomaly; otherwise, it offers recommendations on how to resolve the anomaly, requiring user intervention. The main contributions of this paper are as follows:

- *ZSM-enabled NMS:* We propose a novel anomaly detection and resolution pipeline to effectively enable trustworthy ZSM in NMSs by leveraging the latest advancements in AI. This pipeline integrates AI|XAI|LLM.
- *Dynamic scaling use case with ZSM:* We implemented the proposed pipeline on a real-world 6G NMS use case to dynamically scale cloud resources for a microservice application. This implementation employed XGBoost [5] for AI, SHAP [6] for XAI, and Llama2 [7] for LLM.
- *Real-world deployment and testing:* The use case was deployed on an edge computing cluster managed by Kubernetes. Llama2 was deployed on a single NVIDIA A100 GPU. Comprehensive real-world tests, including multiple LLM evaluations, were conducted to optimize performance for this specific use case.

The remaining sections of this paper are structured as follows: Section II describes related works and background. In section III, we illustrate the pipeline-based system's architecture and use case. Section IV showcases the demonstration setup and results. Section V presents future research directions. Finally, section VI concludes the paper.

II. Related works and Background

In this section, we briefly overview related works and the background of each pipeline component. Then, we present the differences between existing solutions and our approach.

A. AI

AI-based methods have been widely used in networking anomaly detection tasks. For example, in [8], the authors proposed an anomaly detection method using the XGBoost classification algorithm to enhance network security by accurately identifying and classifying traffic anomalies. XGBoost [5] is a powerful AI technique that constructs an ensemble of decision trees to achieve high accuracy in classification tasks. However, these AI methods' lack of interpretability is a significant limitation. These models are often considered black boxes because they provide predictions without transparent insights into their decision-making process, making it difficult to understand the underlying causes of anomalies. As a result, XAI methods have been employed to provide interpretable explanations for the predictions made by these AI models.

B. XAI

Several XAI techniques exist and can be classified into global (explaining the entire model) or local (explaining specific predictions). One popular choice is SHapley Additive exPlanations (SHAP) [6], which functions as both a local and global XAI method. It operates by decomposing the output of a model into the sums of the impact of each feature, thereby calculating a value that represents the contribution of each feature to the model's outcome. These values are used to understand the importance of each feature and explain the model's results. SHAP is widely utilized in the literature, particularly in the context of root cause analysis.

C. LLMs

LLMs are being developed to support a wide range of tasks, including text generation, machine translation, question answering, and information retrieval [4]. Although LLMs can achieve high performance on many Natural Language Processing (NLP) tasks, they are not explicitly designed for network management tasks. Two main approaches are used to adapt LLMs to other specific domains: supervised fine-tuning and in-context learning [4]. The first one involves refining the model's performance by training it on specialized datasets, while the second (e.g., zero-shot and few-shot learning) is an alternative approach by giving the knowledge in the input (prompt) without altering the LLM's weights. This adapts the LLMs to new tasks without requiring the fine-tuning computation complexity.

D. XAI & LLMs vs Our approach

Recent developments have extended the focus of XAI towards LLMs, wherein multiple strategies are employed to integrate XAI with LLMs. These strategies are explained in [9]. The 9th strategy in the latter concerns using LLMs to generate user-friendly explanations for XAI outputs. Our work builds upon this strategy by further exploiting LLMs' reasoning capabilities for anomaly resolution [4]. Additionally, some research works utilize LLMs for anomaly detection and explanation [10]. Furthermore, the authors of [11] proposed creating Large Multi-Model Models (LMMs) specialized for wireless systems, enabling dynamic network adaptation and improving logical reasoning. These models can be applied to use cases such as reasoning about anomalies in the network. However, they will rely solely on LLMs for anomaly detection and resolution, which leads to triggering the LLMs more frequently, resulting in significant energy consumption. Our approach, on the other hand, only requests the LLM when anomalies are detected. This is achieved by employing smaller AI and XAI modules to handle KPI reasoning, thus effectively mitigating the energy consumption problem.

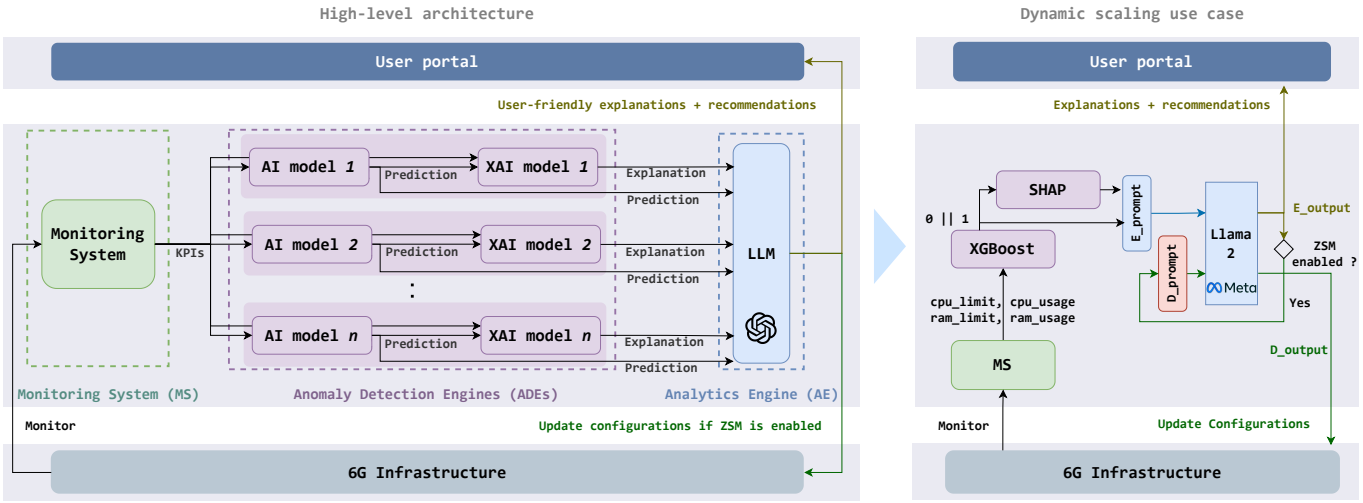

Fig. 1: LLM-enabled trustworthy ZSM architecture design and use case.

III. System Design

In this section, we present the high-level architecture of the ZSM framework and illustrate its practical application through a use case involving the scaling of cloud resources (i.e., CPU and RAM) for 6G microservices.

A. High-level architecture

The ZSM framework features a closed-control loop for managing 6G services, as illustrated in Fig. 1. It consists of three layers: *(i) 6G Infrastructure,* which includes cloud/edge clusters and radio units supporting 6G services. It encompasses components like the 6G Core Network, RAN cloud solutions, and wireless base stations; *(ii) NMS Anomaly Detection and Resolution,* this module autonomously detects and resolves anomalies in the infrastructure, such as issues in base stations or cloud clusters. It operates above the infrastructure layer; *(iii) User Plane,* this top layer involves users deploying 6G services and interacting with the NMS. For instance, users deploying XR services may require a core network and a set of edge applications with specific SLA latency and throughput to support XR requests. These users also receive human-like explanations from the NMS regarding detected anomalies, recommendations to resolve them, or actions taken by the NMS if ZSM is enabled.

As shown in Fig. 1, the design of the NMS consists of three stages: *(i) Monitoring System (MS),* the MS collects KPIs from the 6G infrastructure, including network latency, packet loss, and resource usage metrics. These KPIs are essential for the ADEs to identify anomalies; *(ii) Anomaly Detection Engines (ADEs),* the ADEs use AI to detect anomalies and XAI to provide numerical explanations. Each ADE focuses on specific anomalies, such as QoS issues or security threats, and uses XAI to explain feature contributions; *(iii) Analytics Engine (AE),* the AE processes outputs from the ADEs, using an LLM to reason about predictions and explanations. It provides comprehensive explanations for root causes and suggests actions. If ZSM is enabled, the LLM can autonomously execute these actions without human intervention.

B. Dynamic scaling use case

This use case ensures that CPU and RAM resources for a given microservice application are dynamically adjusted to prevent SLA latency violations. For this purpose, we employed XGBoost as the AI model, SHAP for XAI, and Llama2 as the LLM. Scaling up is performed using the pipeline as illustrated in Fig. 1, whereas scaling down is implemented using a simple heuristic. Below, we describe the scaling-up process, and for more information about scaling down, readers can refer to [3].

1) XGBoost as AI: As depicted in the right-side of Fig. 1, XGBoost receives cloud monitoring KPIs from the MS, i.e., CPU and RAM-related information. This data predicts whether the application will violate the SLA latency. To achieve this, we trained it using the dataset of a microservice application presented by *Mekki et al.* [12]. First, we labeled the dataset's entries as SLA latency violated (y=0) or not (y=1) when the response time is lower or higher than a given threshold. In our case, and based on the experimental data from the dataset, we determine the threshold by measuring the application's latency when it is allocated more resources than required. This value represents the optimal response time that the application can achieve. Then, training is performed using CPU and RAM information as inputs, with SLA violation labels as the output.

2) SHAP as XAI: The XAI module of the ADE relies on a local explanation method based on SHAP to calculate feature importance scores that influence XGBoost's output. Negative values indicate that a feature drives the model's output towards 0, while positive values indicate that a feature drives the output towards 1. For example, if the CPU usage has a score of -2.56, it means that the CPU usage value pushed the model towards output 0 (SLA not respected) with a score of 2.56. Conversely, if the RAM limit has a score of +0.99, this indicates that the feature pushed the model towards output 1 (SLA respected).

3) Llama2 as LLM: Llama2 is employed as the AE. This model is trained using in-context learning, where knowledge is injected into the prompts. From Fig. 1, we can see two prompts corresponding to two LLM tasks. *E_prompt* contains information on how SHAP operates, alongside the values

of XGBoost and SHAP. The objective is to prompt Llama2 to employ deductive reasoning to interpret these values and provide a human-readable explanation to the user, along with recommendations on how to mitigate the SLA violation. Deductive reasoning is used here because it involves applying general rules to specific instances to draw conclusions, i.e., given SHAP values, Llama2 can apply implicit rules to deduce the root causes and suggest appropriate mitigation strategies. The reasoning can be expressed through the following rules: (i) *XAI values* $\Rightarrow$ *deduce the root cause*; and *(ii)* *x is the root cause* $\Rightarrow$ *increase the limits of x*. For example, if the SHAP values show that CPU usage is a significant factor in SLA violations, the model can deduce that increasing the CPU is necessary. Secondly, if ZSM is enabled, *D_prompt* provides the *E_output* with some instructions to Llama2 to perform Named Entity Recognition (NER) and extract the values in a JSON format acceptable by the infrastructure (*D_output*). We opted for task decomposition into two tasks (i.e., first reasoning to generate *E_output*, then NER to extract *D_output*) because combining them resulted in more generation errors.

IV. Performance Evaluation

In this section, we first describe the evaluation setup. Following that, we outline three evaluation steps: *(i)* one for the entire framework, where we present the dynamic scaling use case results; *(ii)* the second for evaluating LLMs' reasoning capabilities; and *(iii)* the third for assessing the trustworthiness of *E_output*. Finally, we present the evaluation conclusion.

A. Evaluation setup

Our experimental setup, illustrated in Fig. 2, includes two machines hosting the Kubernetes-based cluster and the Llama2 LLM, respectively. The first machine, equipped with an Intel(R) Xeon(R) Silver 4314 CPU (2.40GHz), runs Ubuntu OS 22.04.3 LTS and Kubernetes to manage a single-node cluster hosting the MS and ADE components. The second machine, with an Intel(R) Xeon(R) Gold 6240R CPU (2.40GHz) and an NVIDIA A100 GPU (40GB vRAM), also runs Ubuntu OS 22.04.3 LTS and NVIDIA CUDA driver version 545.23.06. It uses Docker to run the Llama2 LLM facilitated by textgen-webui. The LangChain framework handles LLM prompt preparation, with parameters set to a maximum of 2000 tokens, a temperature of 0.1, and a repetition penalty of 1.15.

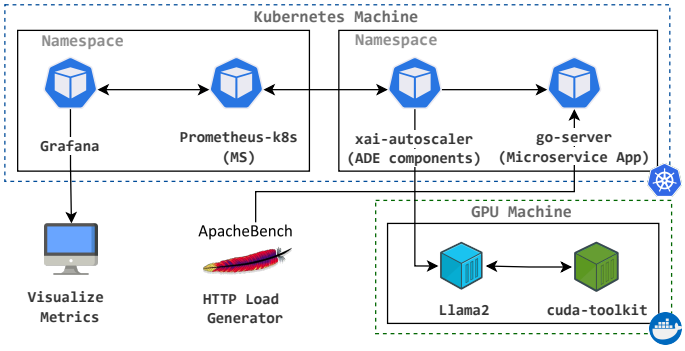

Fig. 2: Evaluation setup.

B. Evaluation: Dynamic scaling framework

1) Scenario: We initially configured the microservice application with 0.25 CPU cores and 256 MB of RAM. To evaluate performance under load, we used ApacheBench to generate high volumes of concurrent HTTP requests. The stress tests involved executing a predefined pattern of requests five times, with varying rounds of concurrent clients (c) and total requests (n): 15 rounds with 50 c and 200 n, 10 rounds with 100 c and 400 n, 15 rounds with 300 c and 500 n, and 20 rounds with 50 c and 50 n. This pattern is illustrated in Fig. 2. Throughout the tests, we monitored CPU and RAM using Prometheus, as well as the LLM's *E_output* and *D_output*. For performance comparison, we executed two additional approaches: the state-of-the-art scaling method based on XAI and heuristic decision-making from [3] (denoted *"xai-heur"*) and a basic heuristic approach without the XAI module (denoted *"no-xai"*), also from [3]. The key distinction is that our approach (*"xai-llm"*) allows flexible resource allocation, while the heuristic approaches use fixed values. Additionally, *"no-xai"* does not use an XAI module to provide the root cause, thus updating all resources (both CPU and RAM) when a violation is predicted.

2) Results: From Fig. 3, it is evident that CPU and RAM allocation adapt dynamically to the application load. For instance, the RAM limit increases with usage during heavier loads, and the CPU behaves similarly. The XGBoost|SHAP|Llama2 pipeline efficiently manages this dynamic scaling (demo: https://youtu.be/1CoDNVWJcqA). Additionally, before updating resources, Llama2 generates explainable, human-like *E_output*. Two examples are shown at the top of the figure at times t_1 and t_2: one for updating RAM and the other for updating both CPU and RAM. This notification is sent to users to enhance trust in the pipeline's decisions. The system then extracts the new configuration from this text and generates *D_output* in a JSON structure, enabling efficient resource updates. However, minimal errors in CPU allocation occurred (e.g., at t_3). Therefore, we evaluate Llama2's role in this pipeline in the next subsection. For more information on XGBoost and SHAP, readers can refer to [3].

In Fig. 4, we compared our approach (*"xai-llm"*) with *"xai-heur"* and *"no-xai"* by calculating the mean CPU and RAM allocations for each (n, c) pair. We found that all three approaches yielded similar latencies but with different resource allocations. The *"no-xai"* method generally allocates more CPU and RAM due to the absence of the XAI module, which results in less efficient resource use. In contrast, our approach prioritizes RAM during dense requests, possibly because the LLM, trained to understand the impact of insufficient RAM on system stability, allocates more to prevent failures. For CPU allocation, our approach generally uses less, except in dense requests where it allocated more than *"xai-heur"*. Averaging these allocations across all (n, c) values, *"xai-llm"* ranks first in CPU allocation, followed by *"xai-heur"*, and then *"no-xai"*. Conversely, *"xai-heur"* ranks first in RAM allocation, followed by *"xai-llm"*, and then *"no-xai"*. Overall, these results demonstrate that the LLM can effectively serve as a decision-maker, competing with state-of-the-art scaling methods while offering user explanations.

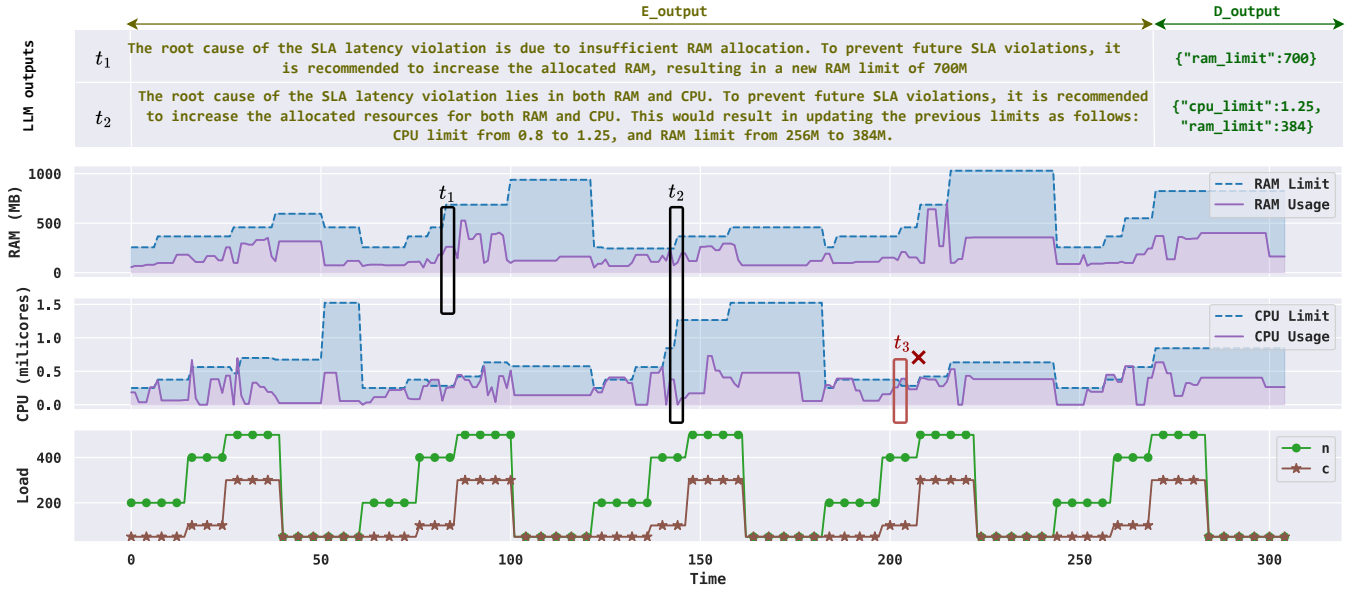

Fig. 3: Dynamic CPU and RAM scaling in response to microservice load.

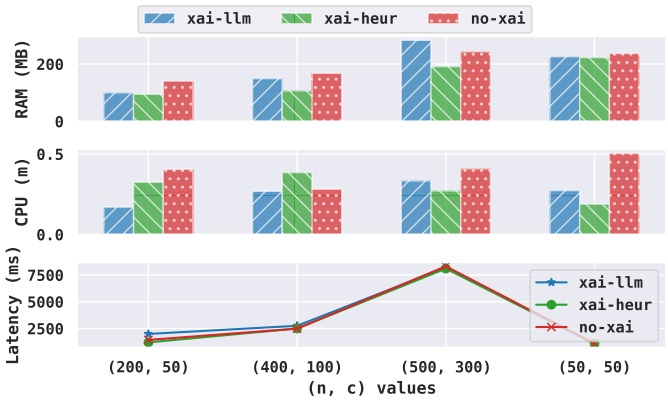

Fig. 4: CPU and RAM allocation comparison.

C. Evaluation: LLM reasoning & decision-making

1) Scenario: In this scenario, we focused exclusively on cases where anomalies were predicted by XGBoost. We generated 100 random allocations for CPU and RAM, with values ranging between 0.25 and 2 cores for CPU, and between 128 MB and 2 GB for RAM. Alongside these allocations, we randomly assigned SHAP values between -5 and 5 to each feature. We evaluated the performance of various open-source LLMs and OpenAI's GPT-4 on generating *E_output* and *D_output*. The LLMs were executed on all the generated samples for evaluation. During this process, we calculated the *Score* of each LLM on this test: +1 if the generated JSON contained the correct new configuration, which entails: *(i)* generating a correct JSON structure; *(ii)* including a feature in the JSON if its SHAP value is negative; and *(iii)* ensuring that the JSON value is greater than the randomly assigned value. A *Score* of +0 was assigned if these conditions were not met. Additionally, we categorized errors made by the LLMs as follows: *(i) JSON errors*, indicating failure to generate a correct JSON structure; *(ii) Amount errors*, where the new CPU or RAM limit was lower than the initial values; and *(iii) Features miss*, where a feature with a negative SHAP value was missing from *D_output*. We also monitored the LLMs' execution times: *Explanation time* and *Update time* for *E_output* and *D_output* generation, respectively.

2) Results: Fig. 5 presents the scores and mean generation times for various LLMs on given tasks, highlighting both *E_output* and *D_output*. OpenAI's GPT-4 achieved the highest score (91/100). Among open-source LLMs, Llama2 70B scored 89/100 with 1 *JSON error*, 4 *Amount errors*, and 6 *Features miss*, making it the best open-source LLM for these tasks. Its execution time is 5.9 seconds (s) for *E_output* and 2.6s for *D_output*, resulting in an E2E time of 8.5 s, longer than most LLMs due to its size. CodeLlama 13B was the second-best with a score of 73/100 and an E2E time of 5s. Vicuna 30B scored 57/100 but had an E2E time of 17s. Phi2, despite being smaller (2B parameters), scored 41/100. Google's Gemma 7B scored 0/100 with 100 *JSON errors*. Overall, Llama2 70B scored highest among open-source models, but its size and E2E time are significant. If these values are critical for other decision-making systems, CodeLlama 13B offers a good balance.

D. Evaluation: LLM's trustworthiness

1) Scenario: In this scenario, we evaluated the semantic quality of the *E_output* generated by each LLM to assess the comprehensibility of the texts and the trust users are likely to place in them. We used several established metrics for this evaluation: *(i) BLEU,* assesses n-gram precision between generated and reference texts, with higher scores indicating better fluency and adequacy; *(ii) METEOR,* measures quality considering exact word matches and semantic similarity using stemming and synonymy, providing a comprehensive evaluation of fluency and semantic fidelity; *(iii) ROUGE-L,*

Fig. 5: LLMs scores and *E_output-D_output* generation times for the dymanic scaling use case.

focuses on the longest common subsequence of words between generated and reference texts, reflecting content overlap and sequence similarity; and *(iv) BERTScore,* evaluates token-level semantic similarity using contextualized embeddings from BERT. Reference texts for these metrics were generated by 10 experts, representing the ideal *E_output*. High scores on these metrics indicate that an LLM produced well-explained and coherent text according to experts' expectations.

2) Results: Fig. 6 illustrates the aforementioned metrics applied to each LLM and sorted from highest to lowest scores. Among the evaluated models, Llama2 with 70B parameters achieves the highest metrics scores, indicating that it consistently produces text semantically similar to domain experts' expectations. Therefore, we can confidently conclude that the *E_output* from Llama2 is trustworthy, enabling reliable ZSM. Surprisingly, Phi2 ranks second in the metrics ranking, which suggests that it also generates high-quality text that is closely aligned with expert expectations. While the *BERTScore* across most LLMs is approximately 0.7, indicating a high degree of semantic similarity with expert texts, the variations in wording choices are evident as shown by *BLEU*, *METEOR*, and *ROUGE-L* metrics. However, overall, nearly all LLMs generate text that closely matches in meaning (high *BERTScore*), albeit with differences in wording. Notably, Llama2 70B stands out as the model-producing text that aligns most closely in wording with user expectations.

E. Evaluation conclusion

We divided the performance evaluation section into three steps. Fig.3 demonstrates a real-world execution of the pipeline, showcasing its efficiency in predicting SLA violations and providing timely updates with human-readable output (demo: https://youtu.be/1CoDNVWJcqA). This illustrates the potential for LLM-based trustworthy ZSM in 6G networks. Given the minimal errors made by the LLMs, we further evaluated the decisions made by these models as depicted in Fig.5. Our analysis identified Llama2 with 70B parameters as the best choice among open-source LLMs, particularly when execution time is not overly constrained. The explanation generation time is 5.9 s, allowing for comprehensive and explainable text generation, while the update time is approximately 2.3 s when ZSM is enabled. Additionally, Fig. 6 showcases the comprehensibility of the generated text using well-known metrics, providing insights into the trustworthiness of these texts. Llama2 ranks highest, indicating it produced text closest to expert expectations. From these explorations, we conclude that LLMs in ZSM can enable trust and powerful decision-making in 6G networks. However, there are some limitations to overcome, which we present in the next section.

V. Limitations and Future Directions

Our work has successfully demonstrated the feasibility of relying on LLMs to enable trust in 6G ZSM frameworks. However, to achieve truly robust LLM-based decision-making, several areas still require future development.

A. Faster LLM inference

From Fig. 5, the main drawback of the LLMs is the long generation time. As this latter is not very scarce in our use case, it can be very important in low-level decision-making, such as in URLLC services. In such use cases, decision time is very important to be close to zero. However, relying on LLMs makes this latter very slow. Therefore, LLM generation time must be investigated to minimize it. Indeed, there are efforts in the literature aimed at increasing LLM inference speed [13].

B. Telecom-aware LLMs

From Fig. 3, LLMs play a key role in the ZSM framework by providing trust and facilitating decision-making. However, these models require training to understand the problem context, relying on in-context learning, which increases the prompt's information content, thus increasing generation time. To address this, the community should focus on developing Telecom-aware LLMs. These models would inherently understand Telecom problems, reducing the need for detailed context information, hence reducing generation time. Additionally, Telecom-aware LLMs can better identify and reason about Telecom-related anomalies across technological domains, an area where generic LLMs struggle without extensive context. While researchers in [14] have presented an initial framework for Telecom-aware LLMs, further research is needed.

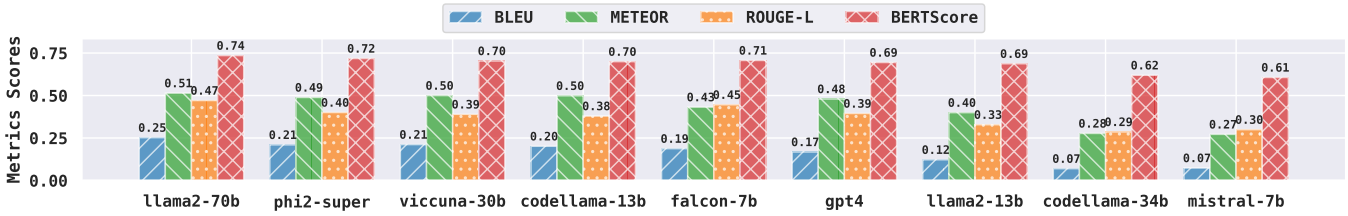

Fig. 6: LLMs metrics score.

C. Green LLMs

From Fig. 5, the best-performing open-source LLM for our use case has 70B parameters. This large model is power-consuming. To address this issue, researchers should focus on developing Small Language Models (SLMs) with fewer parameters (e.g., 1B) that can match the efficiency of their larger counterparts. SLMs require significantly less power, making them environmentally friendly. Although the quality of current SLMs is continuously improving, they are not yet capable of fully replacing larger models. Therefore, the community should invest in the development of SLMs. Additionally, the power consumption challenge of LLMs can be tackled through advanced energy-efficient techniques [15], which should also be further investigated to reduce power usage.

VI. Conclusion

In this paper, we presented a comprehensive framework for achieving trustworthy ZSM in 6G networks. Our approach integrates AI for anomaly detection, XAI for root cause analysis, and LLMs for generating user-friendly explanations and implementing corrective actions. The performance evaluation demonstrated the pipeline's effectiveness in predicting and addressing SLA violations, delivering timely updates and human-readable explanations that enhance user trust.

Acknowledgment

This work is partially supported by the European Union's Horizon Program under the 6G-Bricks (Grant No. 101096954) and the 6G-Intense (Grant No. 101139266) projects.

References

[1] Anon, "Zero-touch network and service management (ZSM); Requirements based on documented scenarios," 2021. [Online] [Accessed: 29-Mar-2021].

[2] S. Wang, J. F. Balarezo, S. Kandeepan, A. Al-Hourani, K. G. Chavez, and B. Rubinstein, "Machine learning in network anomaly detection: A survey," *IEEE Access*, vol. 9,152379–152396, 2021. DOI: 10.1109/ACCESS.2021.3126834

[3] M. Mekki, et al., "XAI-Enabled Fine Granular Vertical Resources Autoscaler," in *Proc. 2023 IEEE 9th International Conference on Network Softwarization (NetSoft)*, pp. 161–169, 2023. DOI: 10.1109/NetSoft57336.2023.10175438

[4] Y. Chang, et al., "A survey on evaluation of large language models," arXiv preprint *arXiv*:, 2023.

[5] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in Proc. 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 785–794, 2016. DOI: 10.1145/2939672.2939785

[6] S. M. Lundberg and S.-I. Lee, A unified approach to interpreting model predictions, vol. 30. 2017, Advances in Neural Information Processing Systems.

[7] H. Touvron, et al., "Llama 2: Open foundation and fine-tuned chat models," arXiv preprint *arXiv*:, 2023.

[8] D. Niu, et al., "A Network Traffic anomaly Detection method based on CNN and XGBoost," in *Proc. 2020 Chinese Automation Congress (CAC)*, pp. 5453–5457, 2020. DOI: 10.1109/CAC51589.2020.9327030

[9] X. Wu, et al., "Usable XAI: 10 Strategies Towards Exploiting Explainability in the LLM Era," arXiv preprint *arXiv*:, 2024.

[10] J. Su, et al., "Large Language Models for Forecasting and Anomaly Detection: A Systematic Literature Review," arXiv preprint *arXiv*:, 2024.

[11] S. Xu, et al., "Large multi-modal models (LMMs) as universal foundation models for AI-native wireless systems," arXiv preprint *arXiv*:, 2024.

[12] M. Mekki, N. Toumi, and A. Ksentini, "Microservices configurations and the impact on the performance in cloud native environments," in Proc. 2022 IEEE 47th Conference on Local Computer Networks (LCN), pp. 239–244, 2022. DOI: 10.1109/LCN53696.2022.9843385

[13] P. Nawrot, et al., "Dynamic memory compression: Retrofitting LLMs for accelerated inference," arXiv preprint *arXiv*:, 2024.

[14] H. Zou, et al., "TelecomGPT: A Framework to Build Telecom-Specific Large Language Models," arXiv preprint *arXiv*:, 2024.

[15] J. Stojkovic, et al., "Towards Greener LLMs: Bringing Energy-Efficiency to the Forefront of LLM Inference," arXiv preprint *arXiv*:, 2024.

BIOGRAPHIES

Abdelkader Mekrache is a PhD candidate at EURECOM's Communication Systems Department. His primary focus is on advanced network management frameworks in next-generation wireless networks under the supervision of Prof. Adlen Ksentini. He is an active participant in collaborative research and notably contributes to the OAI project, as well as multiple European projects, including 6G-Bricks, 6G-Intense, and Sunrise-6G.

Mohamed Mekki obtained his PhD from EURECOM, where he now works as a researcher. He specializes in the Cloud Edge Computing Continuum, focusing on utilizing containerization and WebAssembly technologies to enhance the automated management of applications, while also considering energy consumption and carbon footprint optimization. His work contributes to several European projects, including AC3, 5GDrones, and MonB5G.

Adlen Ksentini is a professor in the Communication Systems Department at EURECOM, leading activities on softwarization, 5G/6G, and edge computing. His research focuses on network virtualization, Software Defined Networking (SDN), and edge computing for 5G/6G networks. He has participated in several H2020 and Horizon Europe projects, including 5G!Pagoda, 5GTransformer, MonB5G, and Sunrise-6G. Currently, he is the technical manager of 6G-Intense and AC3, working on zero-touch management and the Cloud Edge Continuum. His expertise includes Markov Chains, optimization algorithms, and Machine Learning (ML). He is also a member of the OAI board of directors, overseeing Core Network and O-RAN activities.

Bouziane Brik is an Assistant Professor at Sharjah University. He has been (still) working on resources management and security challenges of 5G/6G network slicing in the context of H2020 European projects including MonB5G, 5GDrones, InDiD, and 5G-INSIGHT. His research interests also include 5G and Beyond networks, Explainable AI, and machine/deep learning for wireless networks.

Christos Verikoukis received his BSc and MSc degrees from Aristotle University of Thessaloniki in 1994 and 1997, and his PhD from the Technical University of Catalonia (UPC), Barcelona, in 2000. He is currently a professor at the University of Patras and an affiliated faculty member with ISI/ATH. He has published over 160 journal papers, 240 conference papers, co-authored 5 books, and holds 4 granted patents. He is the EiC of IEEE Networking Letters and a member of the IEEE ComSoc GITC. He has coordinated 20 EC and nationally funded projects.