AICOM: A Novel LLM-based Autonomous Configuration Management in Cloud-Native 5GC

Vitor Kussler Veronese*‡, Zhiyi Zhang‡, Jean-François Maudoux*, Ilhem Fajjari*, Nadjib Aitsaadi‡
*Orange Innovation, F-92320, Châtillon, France
‡Université Paris-Saclay, UVSQ, DAVID, F-78035, Versailles, France
Emails: ⟨firstname⟩.⟨lastname⟩@{*orange.com, ‡uvsq.fr}

***Abstract*—The shift to cloud-native 5G core (5GC) networks has introduced new challenges for configuration management, demanding higher automation, flexibility, and reliability. This paper presents a unified architecture for Automated and Intelligent COnfiguration Management for cloud-native 5GC (AICOM), integrating deterministic automation frameworks, Kubernetes-based orchestration, and Large Language Model (LLM)-enabled agents. Our approach separates Intelligence Layer understanding from Network Automation Layer execution, leveraging GitOps and Artificial Intelligence (AI)-driven intent processing to automate and standardize configuration workflows. It also minimizes the dependency on agents seen in current state-of-the-art intelligent architectures for configuration management by implementing a hybrid approach to task orchestration. We deeply evaluate our proposed solution on a real testbed, demonstrating reliable automation, strong performance across diverse operator intents, and robust recovery mechanisms. While the hybrid model effectively balances deterministic reliability and agentic flexibility, our results reveal current limitations in LLM intelligence and adaptability for novel demands. These findings highlight the need for further advances in LLM architectures and standardization to fully realize autonomous, scalable configuration management in next-generation 5G networks.**
The proposed approach is currently under discussion within the 3GPP SA5 standardization group.

***Keywords*—5GC, Configuration Management, Automation, Network Orchestration, AI-driven Management, LLM, Intent-based Networking**

I. Introduction

The deployment of 5G networks and the anticipated advent of 6G bring unprecedented complexity to telecommunication systems, especially due to the adoption of cloud-native architectures requiring novel management approaches. A 5G network no longer consists of a few monolithic elements but rather hundreds of distributed, containerized microservices running across cloud infrastructures. Ensuring that these myriad of components are correctly configured and work in a coherent way is extremely challenging. In a recent study of TMF Forum [1], about 72% of operators expressed that major outages in the past three years could have been prevented with better configuration management. For example, a single misconfiguration triggered an 86-hour outage in 2022 that affected 39 million subscribers in Japan [1]. These statistics and example illustrate that as networks become more complex, the margin for configuration error shrinks, and the need for automated, reliable configuration management becomes absolutely critical.

In this context, industry initiatives led by 3GPP, ETSI, and the TM Forum are aligning toward the goal of achieving higher levels of network autonomy in 5G and beyond. 3GPP's SA5 group has modernized management and orchestration frameworks to support cloud-native, service-based operations, while ETSI's Zero-touch Network and Service Management (ZSM) initiative envisions self-configuring, self-healing, and self-optimizing networks through closed-loop automation. Building on these foundations, the TM Forum's Autonomous Networks framework defines five levels of autonomy (0–5), from manual operations to fully autonomous, intent-driven networks.

As an example, Orange Telco operator, together with many global operators, has endorsed the TM Forum Autonomous Networks Manifesto and is targeting Level 4 autonomy by 2025–2027, where networks can make, validate, and execute most operational decisions with minimal human intervention.

We recall that configuration management refers to the processes and tools used to track, control, and maintain the consistency of network configurations, ensuring that a system's intended state is always i) known, ii) reproducible, and iii) correct. While this has long been a cornerstone of network operations, its complexity has grown exponentially in the cloud-native 5GC environment. In 5G, network functions (*e.g.*, Session Management Function (SMF), User Plane Function (UPF)) are decomposed into microservices that continuously scale, restart, or migrate across distributed clusters. This dynamic behavior multiplies configuration parameters and interdependencies, which are scattered across Kubernetes manifests, Helm charts, and vendor specific files. As a result, ensuring coherence between the intended and actual network state has become increasingly difficult, with configuration drift emerging as a critical operational risk.

Unfortunately, traditional manual or ad hoc approaches are no longer sustainable under these conditions. Declarative and version controlled methodologies, such as GitOps, become therefore essential to preserve consistency, traceability, and auditability in rapidly changing environments. However, deterministic automation alone cannot fully address the semantic and contextual challenges of 5G management, such as interpreting high level intents, resolving dependencies, or adapting to unforeseen configuration states. These aspects require adaptive intelligence capable of reasoning across multiple layers of abstraction.

This paper presents AICOM, an architecture designed to address the aforementioned challenges in managing the growing complexity of cloud native 5GC configuration and to advance toward automated and intelligent management. It makes three main contributions. First, it introduces a hybrid orchestration framework that deliberately balances deterministic automation, providing reliability, control, and auditability, with agentic LLM driven reasoning to enhance adaptability and contextual understanding. In AICOM, agents are not included as a generic

978-3-903176-82-9

intelligence layer but are activated only when relevant, specifically when deterministic logic alone cannot resolve ambiguity, contextual dependencies, or novel configuration demands. This selective use ensures that intelligence remains purposeful, explainable, and aligned with operational goals. Secondly, it presents a secure and comprehensive LLM integration into a 5GC management system, having LLMOps considerations and integrated tracing between configuration operations and AI components. Finally, AICOM implements an LLM enabled configuration management process that supports natural language intent translation, automated configuration generation, and continuous improvement through operational feedback, paving the way for context aware and self evolving network management. Based on extensive experiments conducted using the Open Air Interface (OAI) 5G core network and several LLM models (Gemini 2, Qwen 3, and Llama 3), the obtained results demonstrate strong performance in terms of reliability and automation capabilities.

The center question of our work is how to leverage LLMs and agentic AI specifically for autonomous configuration management in cloud-native 5GC while minimizing the non-deterministic risks linked with wide AI use, paving the way toward self-managing and self-optimizing 5G and future 6G networks. This work has lead to the registration of two patents in regards to intelligent configuration management and has actively been discussed in the standardization group SA5 of 3GPP.

The rest of this paper is structured as follows. Section II reviews existing architectures, agent-based systems, and standardization efforts. Section III introduces our proposed solution AICOM, including its components, workflows, and integration logic. Section IV presents the experimental setup, while Section V presents the results used to evaluate the system's performance. Finally, Section VI summarizes the key findings and outlines future research directions.

II. Related Work and Challenges

Configuration management in current 5GC networks faces growing challenges related to operational efficiency, scalability, and reliability. As network infrastructures evolve toward cloud-native, declarative, and AI-driven architectures, traditional configuration management paradigms, generally manual and imperative, are increasingly unable to cope with the rising complexity, dynamicity, and heterogeneity of modern deployments.

A. Standardization Landscape and Gaps in 5G Configuration Management

The standardization landscape for configuration management in 5G is primarily shaped by 3GPP and ETSI. While 3GPP's frameworks have long provided a solid foundation, they are increasingly challenged by the evolution toward cloud-native and intent-driven paradigms. As discussed in TR28.834TM [2] and TR28.869 [3], 3GPP is currently evaluating three main evolutionary paths: leveraging ETSI NFV-IFA OAM functions [4], integrating ETSI NFV MANO, or incrementally updating its own management systems. These directions represent an important step forward, introducing more generic OAM functions that better align with a cloud-native-oriented perspective.

In parallel, the TM Forum defines progressive levels of network automation, with the long-term goal of achieving fully autonomous, intent-driven networks [5]. This convergence between telecom and IT frameworks underscores the need for standardized, and adaptive management architectures.

Despite these ongoing efforts, several key limitations persist. Current 3GPP and ETSI approaches only partially address emerging requirements for configuration management, particularly regarding standardized and vendor-neutral data models, integration of AI-driven functionalities, and alignment with *de facto* cloud-native standards such as Kubernetes and GitOps [6].

Several practical challenges persist in today's implementations: i) manual and error-prone processes, ii) complex and interdependent configuration files, iii) limited operational feedback, iv) vendor lock-in, and v) fragmented and difficult-to-evolve configuration databases that hinder scalability across multi-vendor environments.

These persistent challenges highlight the need for new architectural approaches that bridge existing standardization efforts with modern cloud-native principles to enable scalable, automated, and intelligent network management.

B. Literature overview

Recent research has increasingly focused on introducing flexibility, automation, and intelligence into Configuration Management (CM) systems [7]–[16]. Among the explored paradigms, agent-based architectures have gained particular prominence. These systems typically employ specialized agents for configuration planning, validation, and enforcement [13]–[18], allowing distributed decision-making and adaptive behavior. Some frameworks further include user-in-the-loop mechanisms to maintain operational oversight and reduce automation risks [19], while others, such as NVIDIA's initiatives, apply these principles within the Radio Access Network (RAN) domain [20].

Beyond agent-based CM, recent work shows that AI is augmenting GitOps and CI/CD to move from static, policy-driven delivery to intelligent, closed-loop operations [21]–[24]. In AI-enabled GitOps, learned validators and LLM assistants analyze proposed diffs against topology and historical incident data to assess impact, predict resource usage, and auto-generate scenario tests for validation. It is then expected that AI will considerably enhance GitOps into dynamic self-adjusting platforms. Differently from the previous case where an AI agent was at the center of decision making, this research direction shows the capability that AI has to enhance the operation of already existing automation systems maintaining a similar structure.

Certain solutions for configuration management in cloud-native environments utilize legacy imperative protocols such as NETCONF and gNMI by incorporating an intermediary layer. One notable example of this approach is SDCIO [25], which acts as a bridge by integrating these traditional protocols within a Kubernetes operator framework.

More recently, the advent of LLMs has introduced new capabilities for configuration management. LLM-based agents can interpret, generate, and translate configuration artifacts expressed in diverse models [26]–[30]. These abilities are particularly relevant as configuration formats diversify across

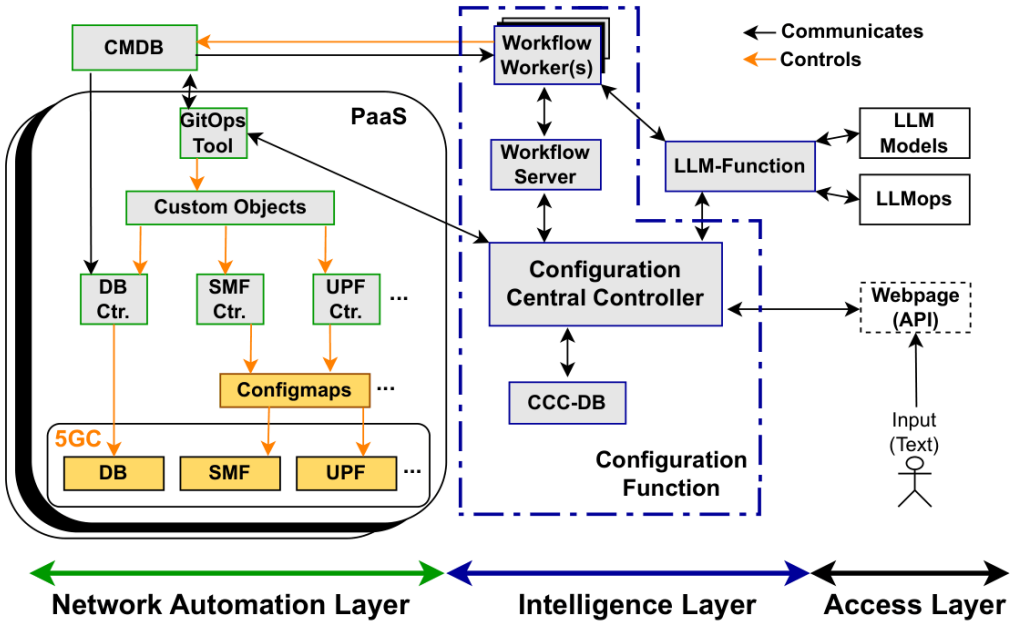


Fig. 1: AICOM: Automated and Intelligent COnfiguration Management for 5GC

5G and 5G+ deployments, enabling LLMs to act as mediators between legacy and cloud-native paradigms. Empirical studies also demonstrate that LLM-driven agents can perform complex operational tasks, such as Linux server administration, with a high degree of autonomy [31].

However, the non-deterministic behavior of LLMs remains a significant limitation for mission-critical telecom applications, where predictability and reliability are essential. To mitigate this, recent studies have proposed deterministic orchestration frameworks that constrain LLM autonomy through structured workflow descriptions [32]. These approaches enhance reproducibility and trustworthiness, while also improving system performance through programmatic workflow representations that support optimization and cost-aware model selection [33]–[35].

Despite recent advances, current approaches to configuration management in cloud-native 5G networks remains limited in several key aspects. Agent-based architectures, which increasingly leverage LLMs for automation and intent processing, often lack deterministic control, resulting in unpredictable and non-reproducible behaviors that are unsuitable for mission-critical telecom environments. Conversely, solutions that integrate imperative components into the cloud-native perspective, such as SDCIO [25], primarily serve as compatibility layers. While they enable basic automation, they generally lack intelligence, adaptability, and the ability to autonomously manage complex or evolving network scenarios. Across the board, existing architectures do not offer intrinsic solutions for end-to-end traceability or scalable management, which are essential as networks grow in size and complexity.

Our proposed architecture directly addresses the key limitations identified in existing approaches. In contrast to agent-based systems that rely solely on LLMs and are therefore prone to non-deterministic behavior [13]–[18], our solution is based on determinism. Configuration operations are orchestrated through structured, auditable workflows that guarantee reliability and reproducibility, as such LLM-enabled agents are invoked only when necessary. Solutions that bridge legacy imperative protocols with cloud-native environments are valuable for managing configuration tasks involving legacy or non-cloud-native components, but they do not provide a complete configuration management perspective [25]. Our architecture can integrate and complement these approaches by adding the intelligence and automation needed for broader, intent-driven and autonomous configuration management.

III. AICOM: Automated and Intelligent COnfiguration Management for 5GC

Our analysis highlights the need for a design that is both modular and adaptive. AICOM (Automated and Intelligent COnfiguration Management) delivers this by weaving together deterministic control, declarative configuration management, and intelligent orchestration within a hierarchical architecture.

AICOM achieves this through a symbiotic integration of three core elements: the 5G core (5GC), automation frameworks, and LLM-based agents. Together, these components create an adaptive and standards-aligned ecosystem capable of overcoming the persistent challenges of manual configuration, limited interoperability, and weak alignment with cloud-native principles.

A. Architecture Overview

AICOM separates concerns into three layers: i) Network Automation Layer, ii) Intelligence Layer, and iii) Access Layer, as illustrated in Fig. 1.

- **Network Automation Layer:** it is the foundation that runs Cloud Native Function (CNF)s on Kubernetes. It turns operator intent into Kubernetes settings and keeps the live system in step with that intent. We use GitOps: a single, versioned Git repository holds all configuration,

and controllers apply changes in push or pull mode. This stable, auditable base is what the Intelligence Layer builds on.

- **Intelligence Layer:** acts as the system's reasoning and control tier, it compiles high-level service intents into runnable workflows spanning multiple automation domains, while maintaining a unified, context-aware network view to enforce policies, preserve traceability, and orchestrate distributed cloud-native components.
  The Configuration Function acts as the semantic and orchestration interface of this layer. It receives configuration intents from the external domain and, in collaboration with the LLM-Function, transforms them into deterministic workflows. This function bridges cognitive reasoning and operational control, maintaining interpretability, validation, and compliance with network policies.
  The central element, the Configuration Central Controller (CCC), manages configuration operations through the Workflow Server and distributed Workflow Workers, ensuring verifiable and fault-tolerant execution. It is the component responsible for the hybrid intelligent control. Contextual and operational data are maintained in the CCC-DB, structured following LLM-inspired memory principles. Short-term memory stores transient execution context and telemetry, while long-term memory retains persistent knowledge such as configuration histories, operational policies, and reference documentation. This organization allows for continuous evolution of the system, as, for example, performance KPIs related to different LLM models may be compared to decide upon the optimal strategy.
  The LLM-Function provides secure interaction with external LLM models. It interprets natural-language intents, assists in configuration synthesis, and validates changes against predefined policies. All interactions are logged to ensure traceability and accountability. Furthermore, it abstracts vendor dependencies by interfacing through standardized APIs and data exchange formats.
  By integrating the Configuration Function and LLM-Function within a deterministic orchestration framework, AICOM achieves a balance between automation reliability and AI adaptability. This design addresses the limitations of traditional 3GPP management systems, which lacks AI reasoning considerations and their integration to deterministic logic.
- **Access Layer:** it represents the interface domain through which human operators, higher-level orchestration systems, or external applications interact with the configuration management framework. This layer forms the entry point for expressing configuration intents, monitoring operational states, and integrating external automation workflows or AI-based assistance. This layer establishes the connection between human intent and the automated execution mechanisms managed by the intelligence and network automation layers.
  In AICOM, the access layer offers a unified programmatic interface that supports intent-driven, human-in-the-loop operation. Operators or external systems can express configuration objectives in natural language or through structured declarative models, which are processed by the Configuration Function. This function collaborates with the LLM-Function for semantic interpretation and reasoning.
  The access layer integrates directly with LLM models. To ensure operational robustness, AICOM adopts an LLMOps-inspired framework, which governs the lifecycle, monitoring, and governance of these AI components. This includes the selection of appropriate LLM models, performance tracking, version control, and the enforcement of access and security policies. By standardizing these interactions, the architecture guarantees controlled, transparent, and auditable use of AI resources across the configuration pipeline.
  Through this integration, the access layer enables adaptive interaction where LLM-based intelligence supports operators without compromising traceability or compliance.
  In doing so, this layer directly addresses the limitations of current 3GPP management frameworks, which still depend heavily on static configuration templates or predefined models, lacking mechanisms for dynamic intent interpretation or adaptive decision-making.

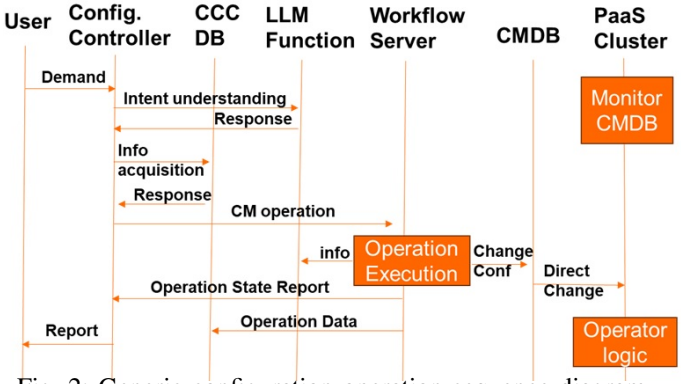


Fig. 2: Generic configuration operation sequence diagram

B. Automation Operation Stage Model

Fig. 2 presents a simplified sequence diagram illustrating how a generic configuration management operation is executed within the AICOM framework. The flow demonstrates the end-to-end process of translating a user intent into a deterministic configuration change applied across distributed cloud-native resources. It involves four main stages: i) intent interpretation, ii) context acquisition, iii) operation execution, and iv) configuration propagation.

At the intent interpretation stage, a user expresses a configuration demand through the external interface using either natural language or a structured declarative format. This request is received by the Configuration Controller, which collaborates with the LLM-Function to infer the underlying technical intent and validate its feasibility. The LLM-Function consults the CCC-DB to retrieve contextual knowledge, historical operations, and relevant policies before generating an interpretable configuration model.

During the context acquisition stage, the Configuration Controller gathers real-time operational data and state information from the Configuration Management Database (CMDB) to ensure alignment between the desired and current network states. This step allows the system to identify potential con-

flicts, dependencies, or constraints before proceeding to the execution phase. It also allows for performance KPIs to be used for better decision-making in relation to the configuration operation (*e.g.*, AI model choice).

The operation execution stage is orchestrated by the Workflow Server, which coordinates workflow tasks across cluster domains. It actively changes the data components in the CMDB to follow the new desired configuration.

Finally, the configuration propagation stage will automatically pass the changes in the CMDB to the CNFs. This is possible due to the constant monitoring of the CMDB, and may take a variable amount of time depending on the interval between monitoring cycles.

Throughout the process, all actions are logged, and both the CMDB and monitoring functions are updated to preserve full traceability.

IV. Performance Evaluation

A. Experimental Environment

To evaluate AICOM, an experimental environment has been developed that includes all components illustrated in Fig. 1. The platform integrates the proposed hybrid LLM orchestration and follows the implementation described in [36].

LLMOps capabilities are integrated via Langfuse [37], supported by a data infrastructure comprising PostgreSQL, ClickHouse, MiniO, and Redis.

The system is designed to support multiple LLMs to enable flexible AI-driven orchestration. In the current implementation, three models are supported: i) Qwen3-32B, ii) Gemini 2.0 Flash, and iii) Llama3-7B.

B. Performance Metrics

To evaluate the effectiveness and efficiency of AICOM, we define the following performance metrics:

T_{conf} **Configuration Operation Time:** The delay between the submission of a natural-language configuration intent to the CCC API and the successful persistence of the generated configuration artifact within the CMDB.

T_{prop} **Configuration Propagation Time:** The delay between the successful persistence of a generated configuration artifact within the CMDB and the same configuration being present in the respective CNF.

S_{conf} **Configuration Success Rate:** The ratio of successful configuration operations to the total number of demands to the API for a configuration operation, reflecting the reliability of the system.

R_{ful} **Configuration Fulfillment Ratio:** The ratio of successfully configured Network Function (NF) to the total number of NFs needed to be configured within a specific operation.

C_{llm} **LLM Budget Cost:** The average monetary cost (in USD) per configuration operation, accounting for LLM API usage. This enables comparison of cost efficiency across different workflow types and LLM models.

M_{ctrl} **Central Controller Memory Consumption:** The total amount of memory (in MB) consumed by the CCC during the execution of a configuration operation.

C_{ctrl} **Central Controller CPU Consumption:** The total amount of CPU resources (in CPU-seconds) consumed by the CCC during the execution of a configuration operation.

Sim **Semantic Similarity:** In the context of text or model output representations, semantic similarity quantifies how similar two texts are based on their vector embeddings.

$$\text{Sim}(\mathbf{A}, \mathbf{B}) = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}} \tag{1}$$

where $\mathbf{A}$ and $\mathbf{B}$ denote the embedding representations of two textual entities.

Ψ_τ **Success Rate Sensitivity to User Intent Variations:** Measures how sensitive a model's performance is to different textual formulations of the same user intent. For a given intent type x and a set of Q semantically equivalent user requests $\rho_1, \ldots, \rho_Q$, the model produces outcome probabilities $p_\tau(y|x, \rho_i)$. The average probability across all variants defines the empirical distribution $p_\tau(y|x)$, from which the sensitivity $\Psi_\tau(x)$ is computed as the normalized entropy:

$$\Psi_\tau(x) = -\frac{1}{\ln C} \sum_{y=1}^{C} p_\tau(y|x) \ln p_\tau(y|x) \tag{2}$$

where C is the number of possible outcomes (e.g., $C = 2$ for success/failure). A higher $\Psi_\tau(x)$ indicates greater variability in model predictions across equivalent intents, reflecting lower robustness to linguistic variations.

The **average sensitivity** across all N intent types is:

$$\Psi_\tau = \frac{1}{N} \sum_{i=1}^{N} \Psi_\tau(x_i) \tag{3}$$

A value of 0 indicates perfect robustness, outcome of the operation may be predicted given the intent group. While a value of 1 shows a lack of knowledge as to the expected result of the operation.

C. Scenarios Description

To comprehensively evaluate AICOM, two complementary scenarios are designed and deployed in our experimental environment. Each scenario targets a distinct aspect of the system's functionality.

*1) **End-to-End Scenario:*** The first scenario evaluates the complete automation process of the AICOM architecture, covering the four stages described in subsection III-B: intent interpretation, context acquisition, operation execution, and configuration propagation. Starting from a configuration intent, the system translates user requests into structured configuration actions, retrieves the required network context from the 5GC, generates the corresponding configuration operations, and propagates them through the framework.

The evaluation focuses on Data Network Name (DNN) configuration management, assessing Create, Read, Update, and Delete (CRUD) operations natively supported by OAI. These capabilities provide a representative and realistic testbed for validating intent-driven configuration automation.

For the operation execution stage, two methods are compared to assess different levels of automation intelligence:

The **zero-shot agent** method executes the task without any prior contextual knowledge or predefined instructions, relying solely on its intrinsic reasoning capability.

The **agent with instructions** method operates with explicit execution guidelines describing how to perform the configura-

TABLE I: Performance comparison

	Deterministic automation	**Zero-shot agent**	**Agent w/ instructions**
S_{conf} (%)	100	0	100
R_{ful} (%)	100	44.31 ± 3.21	100
C_{llm} ($)	0.003 ± 0.000002	—	0.005 ± 0.0002
T_{conf} (s)	58 ± 0.5	—	80.6 ± 27.2
T_{prop} (s)	70 ± 5.7	—	70 ± 5.7
C_{ctrl} (cpu-s)	32.5 ± 0.3	—	34 ± 1.9
M_{ctrl} (MB)	57 ± 1.8	—	29 ± 0.8

tion task, while remaining independent of external information retrieval mechanisms (*e.g.*, RAG [38]).

All methods leverage the *Gemini2* model as the underlying LLM. Each configuration is executed 100 times. The configuration propagation stage is performed identically across both methods, thus eliminating the impact of the operation execution methods.

*2) **Intelligence-Layer Intent Interpretation and SoT Management Scenario (Intent-to-SoT)**:* This scenario assesses the capability of the AICOM architecture to interpret user intents and manage configuration knowledge for complex requests at the Intelligence layer, which corresponds to the first three stages of the automation process: intent interpretation, context acquisition, and operation execution.

Each input consists of a natural language configuration intent, which is semantically interpreted, the configuration operation is then decided upon and executed, resulting in configuration artifacts, that are stored in the Source of Truth (SoT) database (CMDB) without applying changes to CNFs (*i.e.*, excluding configuration propagation stage).

A 5G network slice creation request is selected as a representative use case for its complexity and wide range of involved system components. A 5G slice corresponds to a logically isolated end-to-end network instance that provides customized resources and functions tailored to specific service requirements (*e.g.*, enhanced mobile broadband, ultra-reliable low-latency communication, or massive IoT). Such an operation involves multiple interdependent parameters and logical relationships that exceed the current functional coverage of OAI frameworks. In this case, the considered network functions include the SMF, UPF, AMF, NSSF, PCF, NRF and subscriber database (database and related UDR/UDM).

By terminating execution at the SoT, this scenario emulates a realistic operational boundary between operators and vendors. It also enables systematic evaluation of the system's ability to interpret diverse intents, generate consistent configuration artifacts, and orchestrate deterministic workflows in a controlled environment. Slice provisioning, as specified by 3GPP, is followed in the tests, but the scope is defined strictly to the 5G core.

For each scenario, the reported values correspond to the mean performance and the associated 95% confidence interval.

V. Experimental Results Analysis

A. Automation Evaluation

For the **end-to-end scenario**, the performance of the automation process across the two configurations is summarized in TABLE I. A deterministic automation without LLM integration is used as the baseline for comparison.

The zero-shot agent configuration fails to perform the DNN update operation, indicating that current LLM capabilities are insufficient for complex configuration tasks that require reasoning across multiple interdependent network functions, it correctly configures on average almost half of the total NFs, but is never capable of fully completing the operation in the realized tests. In contrast, the agent with instructions achieves a 100% operation success rate, confirming that providing task-specific guidance enables the LLM to execute the same configuration successfully. However, this configuration exhibits higher configuration latency (T_{conf}) and considerable LLM cost (C_{llm}) due to the additional exchanges required between the controller and the LLM when compared to a deterministic implementation. Interestingly, memory consumption (M_{ctrl}) decreases by nearly 50% compared with the deterministic workflow system, as the agent dynamically generates and resolves intermediate steps instead of storing explicit workflow definitions in memory.

Regarding the configuration propagation stage (*i.e.*, from the CMDB to the 5GC CNFs), the results show identical response times compared to the baseline. This proves the separation and independence of the layers, as the intelligence methods do not affect the configuration propagation time.

Simpler configurations demands, such as single NF change, have shown a high success rate in the zero-shot case. More complex demands show difficulties in the configuration of specific functions more often than others. As an example, for slice creation fewer errors have been observed regarding the NSSF, SMF and UPF, while the configuration of the UDR and the related subscriber database (DB) was more challenging for the AI component.

These results confirm that AICOM provides reliable end-to-end automation by combining deterministic execution with LLM-based adaptability, while ensuring stable configuration propagation behavior. They also prove that the utilization of LLMs for higher levels of autonomy is possible, but introduces trade-offs, particularly in terms of increased resource consumption. The LLM utilization needs to be carefully considered, as a direct integration into the system (zero-shot) does not present acceptable results for the configuration management requirements.

B. Intent understanding test

While the first scenario validates the end-to-end automation workflow, the **Intent-to-SoT scenario** focuses on evaluating the system's capability to understand and process user intents expressed in natural language.

We investigate the impact of intent formulation on automation performance, examining how the structure, length, clarity, and complexity of user inputs influence the generated configuration artifacts and success rate.

To capture the variability of natural language expression in user intents, 10 representative input types are defined:

Type 1: Short and clear
Type 2: Longer with context
Type 3: Very long and detailed
Type 4: Technical jargon-heavy
Type 5: Conversational and informal
Type 6: Extremely verbose and redundant
Type 7: Formal and business-like
Type 8: Command-like (similar to CLI instructions)
Type 9: Unclear or ambiguous

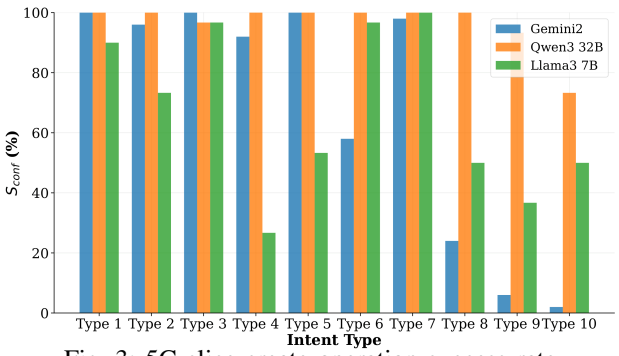


Fig. 3: 5G slice create operation success rate

Type 10: Questioning and uncertain

Although the classification is manually defined, the distinctiveness of these types is validated through a semantic similarity analysis. The semantic similarity values are computed based on BERT [39] sentence embeddings, averaged across at least 15 representative configuration operation demands.

We compared the average pairwise similarity within each type (intra-group) to the average similarity across different types (inter-group). The intra-group similarity averaged 0.935 ± 0.023, while the average of the maximum similarity from any sample of a type to all other types was 0.854 ± 0.018.

The relatively high inter-group similarity is expected, as all intents target the same configuration objective (*i.e.*, 5G slice creation). In contrast, the consistently higher intra-group similarity confirms that the manually defined intent categories are semantically coherent and well separated.

With the intent groups validated as semantically distinct, we evaluate the intent understanding capability of the system. We consider both the impact of different input types and the influence of the LLM model itself. The success rates of the 5G slice creation operation across the three different models and ten input types are presented in Fig. 3.

As shown, when the user input is formulated in a clear, detailed, and formal manner, the LLM is more likely to understand the user intent and generate the correct configuration. Conversely, when the request is unclear, ambiguous, or uncertain, obviously, the LLM models tend to struggle with user intent interpretation.

The success rate depends on the model. The smallest model, Llama3 7B, achieves the lowest success rates over clear intent types ([type 1-type 5]). This behavior can be expected, as it is the most lightweight model among the three. Gemini2 and Qwen3 32B have comparable success rates across most input types. However, when user inputs are redundant, uncertain, or ambiguous, Qwen3 32B achieves a significantly higher success rate than Gemini2.

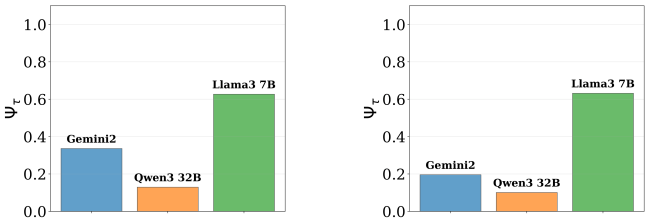
To investigate the reason behind the lower success rates of Gemini2 and Llama3 7B compared with Qwen3 32B, we analyze the failure cases. The failed outputs were categorized into four error types:

Error 1 (E1): Demand for more information
Error 2 (E2): Wrong intent classification into operation
Error 3 (E3): Configuration generated with wrong structure
Error 4 (E4): Configuration generated not respecting intent (*e.g.*, wrong fields or inconsistency between generated files)

The proportion of the error types, averaged across all input types, is presented in TABLE II.

TABLE II: Error rates for all intent types

	Gemini2	Qwen3 32B	Llama3 7B
S_{conf}	66.3%	96.3%	67.3%
E1 rate	30.2%	0.7%	3.3%
E2 rate	0.4%	2.6%	4.7%
E3 rate	0%	0.4%	0%
E4 rate	3.1%	0%	24.7%



(a) Direct success consideration (b) Ambiguous cases consideration

Fig. 4: LLM sensitivity over intent types

For Gemini2, it can be observed that 30.2% of failed cases correspond to error type 1, meaning that Gemini2 requires additional clarification from the user before processing, instead of executing the configuration operation. In contrast, Qwen3 32B tends to infer and generate a configuration even when the user input is poorly structured, for example, uncertain or ambiguous. Thus, Qwen3 32B can handle more natural language expressions, but it also increases the chance of misunderstanding the user intent. The proportion of error 2, wrong intent, is more than six times higher for Qwen3 32B than for Gemini2.

It is important to note that, from a 5GC operator's perspective, performing no action is preferable to executing an incorrect configuration. Furthermore, since the users of our system are network administrators, they are expected to refine their inputs when they are notified that the initial intent is unclear. With better-structured or more detailed input in response to such system feedback, the success rate with Gemini2 can exceed 96%, which is comparable to that of Qwen3 32B.

Small variations in the user input given to a LLM may lead to significant differences in its output [40]. From the perspective of mobile network operators, predictability of the system is important. To quantify how consistently the proposed system reacts to small changes in an intent description within one of the intent categories, a sensibility metric Ψ_τ is considered. A lower value of Ψ_τ indicates higher predictability, meaning that the system provides consistent outcomes even when the phrasing of user inputs varies. Higher values indicate that small changes in user phrasing can cause inconsistent behavior. Therefore, lower Ψ_τ values are desirable, as they imply more stable and reliable system responses.

Fig. 4a presents the average sensibility metric Ψ_τ for each LLM (Gemini2, Qwen3, and Llama3), computed as the expectation over all intent groups for the configuration operation 5G slice creation.

Obviously, Llama3 7B exhibits the highest sensibility, indicating that it is more unpredictable than the other models. This is expected, as its parameter size is the smallest. Qwen3 32B exhibits the lowest value of Ψ_τ, showing that small variations in intent formulation have little impact on its responses. Operations that are more likely to succeed tend to do so consistently, while those that tend to fail usually fail as ex-

pected. Gemini2 presents higher unpredictability than Qwen3. This can be attributed to its frequent requests for clarification when processing unclear inputs, which are considered failures in the baseline case. When such clarification requests are instead considered successful, as shown in Fig. 4b, Gemini's predictability improves. However, its performance remains below that of Qwen3.

Overall, well-structured and explicit intents generally give more accurate interpretations than ambiguous or uncertain ones. Both Gemini2 and Qwen3 32B can achieve success rates above 90%. Gemini behaves conservatively, frequently requesting additional clarification, while Qwen3 proceeds even under uncertainty. Qwen3 has a better success rate but occasionally generates incorrect configurations with wrong intent understanding. We highlight that a certain degree of human confirmation remains necessary to ensure reliable configuration management, and that having such metrics and analysis linked to the management system will greatly improve the predictability and overall use of LLM models in regards to intent recognition and task execution.

C. Workflow learning test

This test consists of passing multiple times a single user intent for slice creation, following the **Intent-to-SoT scenario**, and observing how well the system learns the new operation requested. The test is executed from an initial state with no prior knowledge and is capped at a maximum of 10 requests or until a successful workflow is generated.

The learning mechanism of the system follows a basic closed-loop approach commonly adopted in such learning-loop deployments [41]–[43]. It consists of four main steps: i) Action, ii) Validation, iii) Reflection, and iv) Learning.

The validation step plays an important role in the learning loop and therefore requires particular attention. In this study, validation is performed under an idealized assumption of complete knowledge of slice configuration rules. Specifically, the full set of configuration constraints is used to assess the generated response, enabling the detection of precise errors that can be exploited for iterative refinement. However, more realistic validation mechanisms remain an important direction for future work, as validation constitutes a critical component of practical closed-loop learning systems.

Reflection was needed to translate errors into actual changes in operation execution, as there is a variety of ways to make mistakes in configuration generation. A considerable example is the hallucination of fields in the generation of a file.

The average configuration fulfillment ratio and the associated LLM cost, both obtained using the *Gemini 2* model, are shown in Fig. 5.

A total of 70% of the tests reached the correct configuration within 10 iterations. Half of the tests converged quickly, taking less than 5 iterations to reach a valid configuration. Beyond this point, convergence remains possible but becomes less likely, as the prompts tend to stabilize and become increasingly instruction-heavy, limiting further effective adjustments.

Diminishing returns can be observed for runs with a high number of iterations. This behavior is illustrated by the average fulfillment ratio shown in Fig. 5 (left). The best-fitting curve of the experimental data is given by $y = 89.3\left(1 - e^{-0.587x}\right) + 5.3$ indicating a logarithmic convergence trend. The model

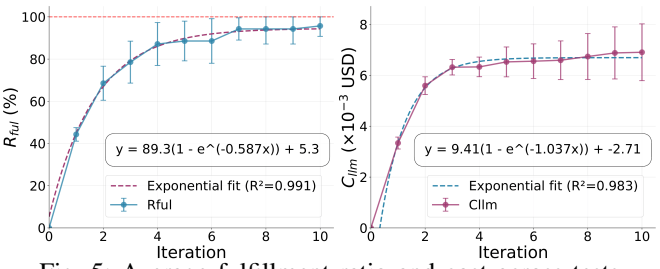


Fig. 5: Average fulfillment ratio and cost across tests

indicates that after the 10th iteration, additional iterations yield only minimal performance gains.

As shown in Fig. 5 (right), the LLM cost increases rapidly during the initial iterations due to the growth of prompt size and the higher number of tool calls. As the learning process progresses and the prompts become more stable, the cost growth slows down and eventually levels off.

One final observation concerns the error analysis, where equal weight was assigned to each configuration object. In fact, some objects may present greater challenges than others when generating a valid configuration. Identifying these bottlenecks could improve learning efficiency and reduce the number of required iterations.

The results indicate that LLMs are capable of learning complex tasks in the context of 5GC cloud-native configuration management. Although convergence is not guaranteed, the probability of obtaining a successful workflow within 10 iterations is approximately 95% per test. These results could be further improved through more advanced learning mechanisms and supporting technologies, such as digital twins. Enriching the current approach with more realistic validation scenarios could provide additional insights into real-world deployment conditions.

VI. Conclusion

This work introduces AICOM, a hybrid architecture for autonomous configuration management in cloud native 5GC networks, combining deterministic automation with LLM driven reasoning. Experimental results confirm the feasibility and robustness of the proposed approach, demonstrating reliable intent interpretation, consistent configuration propagation, and efficient resource utilization. The hybrid orchestration model, merging structured workflows with adaptive intelligence, proves effective for complex operations while revealing current limitations in LLM contextual reasoning and autonomy.

The capability of AI to learn configuration workflows related to complex cloud-native 5GC has been demonstrated. However, further evaluation with stricter conditions and more realistic validation scenarios is required before real-world implementation.

Our ongoing work focuses on extending reasoning capabilities by enabling collaborative multi-agent decision-making. It also addresses cost–performance trade-offs, scalability, multi-vendor scenarios, and the execution of closed-loop tests under more realistic conditions. In parallel, these developments are being promoted within the 3GPP SA WG5 Management.

References

[1] TMForum, "The role of configuration management in 5G evolution," https://inform.tmforum.org/features-and-opinion/

the-role-of-configuration-management-in-5g-evolution, accessed: 2025-10-25.
[2] 3GPP, "3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Study on management of cloud-native Virtualized Network Functions (VNF) (Release 18)," 3GPP, Tech. Rep. TR 28.834 V18.0.1 (2023-06), 06 2023.
[3] ——, "3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Study on cloud aspects of management and orchestration(Release 19)," 3GPP, Tech. Rep. TR 28.869 V1.4.0 (2025-04), 04 2025.
[4] ETSI, "Network Functions Virtualisation (NFV) Release 5; Architectural Framework; VNF generic OAM functions and other PaaS Services specification," ETSI, Tech. Rep. ETSI GS NFV-IFA 049 V5.2.1 (2024-12), 12 2024.
[5] A. R. E. Boasman-Patel, D. Sun, Y. Wang, C. Maitre, J. Domingos, Y. Troullides, I. Mas, G. Traver, and G. Lupo, "Autonomous Networks: Empowering Digital Transformation For The Telecoms Industry," TM Forum, Tech. Rep., 05 2019.
[6] ETSI, "Network Functions Virtualisation (NFV) Release 5; Architectural framework; Report on VNF management gap analysis with open source projects ," ETSI, Tech. Rep. ETSI GR NFV-IFA 051 V5.1.1 (2023-10), 10 2023.
[7] L. Lu, C. Liu, C. Zhang, Z. Hu, S. Lin, Z. Liu, M. Zhang, X. Liu, and J. Chen, "Architecture for Self-Evolution of 6G Core Network Based on Intelligent Decision Making," *Electronics*, vol. 12, no. 15, 2023. [Online]. Available: https://www.mdpi.com/2079-9292/12/15/3255
[8] M. Corici, E. Troudt, P. Chakraborty, and T. Magedanz, "An Ultra-Flexible Software Architecture Concept for 6G Core Networks," in *2021 IEEE 4th 5G World Forum (5GWF)*, 2021, pp. 400–405.
[9] Q. Yu, J. Ren, H. Zhou, and W. Zhang, "A Cybertwin based Network Architecture for 6G," in *2020 2nd 6G Wireless Summit (6G SUMMIT)*, 2020, pp. 1–5.
[10] Y. Li, J. Huang, Q. Sun, T. Sun, and S. Wang, "Cognitive Service Architecture for 6G Core Network," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 10, pp. 7193–7203, 2021.
[11] Z. Duan, G. Nan, R. Li, Z. Wang, L. Xiong, C. Yuan, G. Liu, H. Xu, Q. Cui, X. Tao, and T. Q. S. Quek, "Agile Orchestration at Will: An Entire Smart Service-Based Security Architecture Towards 6G," 2025. [Online]. Available: https://arxiv.org/abs/2505.22963
[12] P. M. Tshakwanda, H. Kumar, S. T. Arzo, and M. Devetsikiotis, "SE-DO: Navigating the 6G Frontier with Scalable and Efficient DevOps for Intelligent Agents Optimization," in *2024 IEEE 14th Annual Computing and Communication Workshop and Conference (CCWC)*, 2024, pp. 0269–0277.
[13] A. Mekrache, A. Ksentini, and C. Verikoukis, "Intent-Based Management of Next-Generation Networks: an LLM-Centric Approach," *IEEE Network*, vol. 38, no. 5, pp. 29–36, 2024.
[14] Q. Li, Y. Xiong, Z. Li, C. Ma, H. Yu, G. Sun, L. Luo, and Z. Zhang, "KlonetAI: Automating (Com)2Nets Management With Human Language Intents," *IEEE Network*, vol. 39, no. 3, pp. 12–19, 2025.
[15] Z. Chen, Q. Sun, N. Li, X. Li, Y. Wang, and C.-L. I, "Enabling Mobile AI Agent in 6G Era: Architecture and Key Technologies," *IEEE Network*, vol. 38, no. 5, pp. 66–75, 2024.
[16] Y. Xiao, G. Shi, and P. Zhang, "Towards Agentic AI Networking in 6G: A Generative Foundation Model-as-Agent Approach," 2025. [Online]. Available: https://arxiv.org/abs/2503.15764
[17] A. S. Araujo, J. M. O. Das Mercês, R. L. Da Silva, A. V. De Alencar, I. F. Passos, M. P. Sousa, M. C. Dias, T. F. Meneses, and D. F. S. Santos, "An Agentic Approach For Dynamic Software-Defined Network Management Using Large Language Models," in *2024 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, 2024, pp. 221–226.
[18] O. G. Lira, O. M. Caicedo, and N. L. S. da Fonseca, "Large Language Models for Zero Touch Network Configuration Management," 2024. [Online]. Available: https://arxiv.org/abs/2408.13298
[19] S. D'Urso, B. Martini, and F. Sciarrone, "A Novel LLM Architecture for Intelligent System Configuration," in *2024 28th International Conference Information Visualisation (IV)*, 2024, pp. 326–331.
[20] NVIDIA, "AI Agent for Telecom Network Configuration Planning," https://build.nvidia.com/nvidia/telco-network-configuration, accessed: 2025-07-13.
[21] M. Baqar, S. Naqvi, and R. Khanda, "AI-Augmented CI/CD Pipelines: From Code Commit to Production with Autonomous Decisions," *ArXiv*, vol. abs/2508.11867, 2025.
[22] S. Prakash and A. Komal, "Architecting Agentic AI for IT Operations: Design Principles for Enhanced Automation and Resilience," *International Journal of Scientific Research in Science, Engineering and Technology*, 2025.
[23] M. Bhamidipati, "AI-Driven Automation and Reliability Engineering: Optimizing Cloud Infrastructure for Zero Downtime and Scalable Performance," *Journal of Computer Science and Technology Studies*, 2025.
[24] R. Zota, C. Bărbulescu, and R. Constantinescu, "A Practical Approach to Defining a Framework for Developing an Agentic AIOps System," *Electronics*, 2025.
[25] P. L. Vanessa Gyger, "Cloud-native Network Controller," Bachelor's Thesis, University of Applied Sciences Campus Rapperswil-Jona, Rapperswil, Switzerland, 2024.
[26] F. Li, H. Lang, J. Zhang, J. Shen, and X. Wang, "PreConfig: A Pretrained Model for Automating Network Configuration," 2024. [Online]. Available: https://arxiv.org/abs/2403.09369
[27] 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, pp. 1–7.
[28] C. Gunawat and A. Khanna, "AI-Enhanced Infrastructure as Code (IaC) for Smart Configuration Management," 2025. [Online]. Available: https://ssrn.com/abstract=5202062
[29] G. Hollósi, D. Ficzere, and P. Varga, "Generative AI for Low-Level NETCONF Configuration in Network Management Based on YANG Models," in *2024 20th International Conference on Network and Service Management (CNSM)*, 2024, pp. 1–7.
[30] A. Dubey, C. P. Singh, and D. Nadig, "Leveraging Large Language Models for Intent-Based Generation of Cloud-Native Configurations," in *2024 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*, 2024, pp. 1–6.
[31] C. Cao, F. Wang, L. Lindley, and Z. Wang, "Managing Linux servers with LLM-based AI agents: An empirical evaluation with GPT4," *Machine Learning with Applications*, vol. 17, p. 100570, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S266682702400046X
[32] B. A. Junaid Syed, Sultan Syed, "Reliability over Unfettered Autonomy: Advocating for Deterministic Orchestration in Large Language Model Tool Integration," *Journal of Computer Science and Technology Studies*, vol. 7, no. 5, p. 989–998, Jun. 2025. [Online]. Available: https://al-kindipublishers.org/index.php/jcsts/article/view/9924
[33] Z. Z. Wang, J. Mao, D. Fried, and G. Neubig, "Agent Workflow Memory," 2024. [Online]. Available: https://arxiv.org/abs/2409.07429
[34] Z. Z. Wang, A. Gandhi, G. Neubig, and D. Fried, "Inducing Programmatic Skills for Agentic Tasks," 2025. [Online]. Available: https://arxiv.org/abs/2504.06821
[35] Q. Zhang, M. Wornow, and K. Olukotun, "Cost-Efficient Serving of LLM Agents via Test-Time Plan Caching," 2025. [Online]. Available: https://arxiv.org/abs/2506.14852
[36] V. K. Veronese, Z. Zhang, J.-F. Maudoux, I. Fajjari, and N. Aitsaadi, "Towards Cloud-Native Configuration Management in the 5G Core Network," Feb. 2026, working paper or preprint. [Online]. Available: https://uvsq.hal.science/hal-05515130
[37] Langfuse, "Open Source LLM Engineering Platform Traces, evals, prompt management and metrics to debug and improve your LLM application." [Online]. Available: https://langfuse.com/
[38] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W. Yih, T. Rocktäschel, S. Riedel, and D. Kiela, "Retrieval-augmented generation for knowledge-intensive NLP tasks," *CoRR*, vol. abs/2005.11401, 2020. [Online]. Available: https://arxiv.org/abs/2005.11401
[39] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2019. [Online]. Available: https://arxiv.org/abs/1810.04805
[40] Z. Zhao, E. Wallace, S. Feng, D. Klein, and S. Singh, "Calibrate before use: Improving few-shot performance of language models," in *Proceedings of the 38th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, M. Meila and T. Zhang, Eds., vol. 139. PMLR, 18–24 Jul 2021, pp. 12 697–12 706. [Online]. Available: https://proceedings.mlr.press/v139/zhao21c.html
[41] W. Lin, J. Wei-Kocsis, J. Zhang, B.-C. Min, D. Gan, P. Asunda, and R. Athinarayanan, "Think, Reflect, Create: Metacognitive Learning for Zero-Shot Robotic Planning with LLMs," 2025. [Online]. Available: https://arxiv.org/abs/2505.14899
[42] Y. Yuan, L. Zhao, W. Chen, G. Zheng, K. Zhang, M. Zhang, and Q. Liu, "Simulating Human-Like Learning Dynamics with LLM-Empowered Agents," 2025. [Online]. Available: https://arxiv.org/abs/2508.05622
[43] H. Zhao, C. Ma, G. Wang, J. Su, L. Kong, J. Xu, Z.-H. Deng, and H. Yang, "Empowering Large Language Model Agents through Action Learning," 2024. [Online]. Available: https://arxiv.org/abs/2402.15809