Agent-as-a-Service: An AI-Native Edge Computing Framework for 6G Networks

Borui Li, Tianen Liu, Weilong Wang, Chengqing Zhao, and Shuai Wang

Abstract

It has become a consensus that the integration of computing, sensing, and communication with ubiquitous intelligence will be the cornerstone of the sixth-generation (6G) network. The concept of edge computing and intelligence, which push the frontier of computation closer to the data source, is a suitable paradigm that aligns with these visions of 6G. In this article, we introduce Agent-as-a-Service (AaaS), an AI-native edge computing framework that leverages AI agents for both the control-plane operations and user-plane services of the 6G network. In the AaaS framework, agents can perform computing, sensing, and communicating tasks automatically by harnessing of the pervasive intelligence offered by 6G infrastructures. By AI-native, we refer to redesigning the whole lifecycle of an edge computing task to align with the prominent reasoning and planning ability of the generative AI. The lifecycle includes plan generation, execution orchestration, resource management, and long-term evolvement. The AaaS framework is built upon emerging techniques such as deviceless computing and WebAssembly to cope with the heterogeneous and geo-distributed 6G edge deployments. Based on the AaaS framework, we conduct a case study on autonomous driving in 6G edge computing to showcase the benefits in terms of overall latency reduction. Finally, we outline potential research directions to form a more efficient and integrated 6G edge computing with artificial intelligence.

Introduction

Recent years have witnessed the emergence and proliferation of edge computing and intelligence [1]. The concept of edge intelligence involves deploying processing and analytics based on artificial intelligence (AI) algorithms at the edge of the network and in closer proximity to the data source, e.g., base stations and dedicated edge servers. Since its inception, there has been a significant surge in edge intelligence applications including smart home, edge video analytics, and the Industrial Internet of Things.

The rapid adoption and evolution of edge intelligence are inseparable from ever-growing support in network architecture. For example, the adoption of virtualized radio access networks (vRAN) in the fifth-generation (5G) networks paves the way for integrating edge intelligence within base stations [2]. 5G vRAN enables running RAN functions on standard commodity hardware, which facilitates sharing the resources of base stations during idle periods with edge intelligence tasks. In turn, the maturity of AI-based network analysis also leads to the standardization of the Network Data Analytics Function (NWDAF) in 5G network architecture.

We now stand at the cusp of transitioning to the sixth-generation (6G) network. As we anticipate the development of 6G, a growing consensus of the omnipresence of AI in future networks [3], [4] and recent advances in AI algorithms motivate us to revisit the network architecture design.

First, the IMT-2030 vision for 6G [5] addresses delivering omnipresent wireless intelligence as one of the key capabilities of 6G networks. To approach this goal of ubiquitous intelligence, simply incorporating AI-based components as network plug-ins is not sufficient. It requires a new network architecture in which AI penetrates each building block of the 6G network, especially at the intersection of 6G and edge computing.

Second, prevalent generative AI such as large language models (LLMs) exhibit strong task fulfillment abilities through advanced reasoning and strategic planning. LLM-based AI agents are reshaping our everyday lives, examples range from copilot programming to robot manipulation. In the realm of networks, recent studies explore replacing existing network components such as network slicing [6], network configuration [7], and air interface management [8] with AI agents. Although these attempts only focus on specific areas, they also reveal the promising potential that both edge services and networks in the future 6G network can be natively operated by intelligent agents.

Motivated by the above observations, we propose an *AI-native* edge computing framework based on AI agents for the 6G network. Unlike prestigious attempts [9] that shed light on network architecture design for better support of AI-based tasks, the AI-native means we redesign the *entire process* of 6G edge computing tasks to harmonize with and fully leverage the advances of generative AI. The redesigned process, i.e., lifecycle, of 6G edge computing tasks starting from task planning and generation to execution orchestration, management, and long-term evolvement. With this

Digital Object Identifier: 10.1109/MNET.2024.3520987
Date of Current Version: 18 March 2025
Date of Publication: 23 December 2024

The authors are with Southeast University, Nanjing 210000, China. Shuai Wang is corresponding author.

AI-native approach, users of 6G edge computing and network maintainers can specify their service intents and let generative AI agents deal with the details of the underlying complicated operations. Hence, we refer to this approach as *Agent-as-a-Service* (AaaS).

Specifically, we draw inspiration from the typical operating system (OS) and design the AaaS framework with two layers: a user-space layer and a kernel-space layer. Akin to the application in traditional OS, agents in user space fulfill the data-plane and control-plane service intents of 6G edge computing. Taking advantage of generative AI's reasoning and planning abilities, kernel-space agents autonomously manage the whole lifecycle of user-space agents with minimal human intervention. Moreover, the foundation model that agents rely on and the tools that implement the necessary operations for agents to invoke reside in an infrastructure layer. This AI-native approach leverages recent advances in agentic AI to minimize the friction to exploit the integrated networking, sensing, and computing capabilities of 6G edge computing.

The contributions of this article are summarized as follows.

- We introduce Agent-as-a-Service, an AI-native edge computing framework for the 6G era. It treats AI agents as first-class citizens to achieve easy-to-use and pervasive edge intelligence. Motivated by the ubiquitous intelligence vision of 6G and the recent algorithm development of LLMs, AaaS leverages AI to manage the lifecycle of agents to reduce manual operation efforts.
- We present challenge issues when applying the AaaS framework to 6G edge computing networks, including adapting to heterogeneous edge devices as well as handling highly dynamic and geo-distributed edge resources.
- We showcase a representative case study with a proof-of-concept implementation of AaaS for autonomous driving. Evaluation results show that AaaS can reduce the average latency of agent execution by 59% and exhibits 20% memory reduction, which is preferred in the edge computing scenario.

Preliminaries and Use Cases

Before we dive into the detailed design of the AaaS framework, we first introduce preliminaries of AI agents, and then we present four representative use cases of AI-native edge computing based on agents.

AI Agents

AI Agent refers to a program or system based on AI models that can react to its surrounding environment by making decisions and taking actions to complete specific tasks [10]. In its early stages of development, AI agents were mainly adopted in the fields of expert systems, which simulate the knowledge and decision-making process of experts. Agents in this period depend on pre-defined rules or procedures, and their generalization and reasoning capabilities are not satisfying especially on out-of-distribution tasks.

With the rise of reinforcement learning (RL), AI agents have been further improved to learn policies autonomously. This progress is facilitated by establishing a pre-defined environment that can provide rewards for the actions of agents. Nevertheless, RL-based agents also have problems such as high demand for training samples and long convergence times.

To approach this goal of ubiquitous intelligence, simply incorporating AI-based components as network plug-ins is not sufficient.

In the past two years, large language models (LLMs) have shown significant advancements in various domains such as instruction understanding, task planning, and engaging in effective natural language interactions with humans. Therefore, LLM-based agents have received much attention for their potential towards artificial general intelligence (AGI). Examples such as programming copilots and embodied AI robots.

Building Blocks of LLM-Based Agents

An identical LLM-based agent consists of four modules: profile, planning, action, and memory [10].

The profile of an agent serves as the initial configuration module for the agent, used to store and manage the agent's basic attributes and functional preferences. For example, a profile can include the agent's task domain, capability range, and configuration of available tools. With the profile, the agent better understands its role and limitations during task execution, allowing for more aligned decision-making in subsequent planning and actions.

The planning module enhances the agent's problem-solving capabilities by decomposing complex tasks into simpler subtasks based on the reasoning ability of LLMs. The component often employs a structured thinking process such as the Chain of Thought (CoT) approach to derivate plans and then dynamically adjusts the plan to align with environmental changes.

Based on the generated plan, an action module is responsible for executing the plans and interacting with the environment. An agent primarily performs actions by invocating various tools such as online searches, database queries, or robot manipulation.

The memory module functions as a repository for storing perceived data from the environment and retaining learned experiences, allowing the agent to accumulate context over time. This accumulation of experience allows the agent to self-evolve, enhancing consistency, effectiveness, and adaptability in its behavior over time.

Use Cases of AI-Native Edge Computing

Embodied AI. It refers to the field of combining AI agents with physical entities or robot systems with physical existence. Recently, researchers have focused on constructing embodied AI agents based on LLMs to plan the actions of robots. The primary function of these embodied AI agents involves controlling mechanical arms and other actuators, which are subject to stringent latency constraints. This scenario underscores the relevance of AaaS-based 6G edge

To make things worse, the generation and operation of agents in AaaS are managed by LLMs, which, due to their resource-intensive nature, cannot be deployed on all the edge devices in the 6G network.

intelligence as a highly suitable paradigm for optimizing the performance of these embodied AI agents.

Autonomous Driving. By combining AI and sensor technology, autonomous vehicles are able to sense, make decisions, and operate autonomously without human intervention. The powerful learning and reasoning capabilities of AI agents can process sensing data, make decisions, and control the state in real time to support requests such as navigation, obstacle detection, and traffic planning. By migrating the agent from 6G edge devices to autonomous vehicles, the driving system reduces the reliance on cloud services and improves response speed to achieve higher efficiency and accuracy.

Smart City. In terms of smart city applications, many sensors are distributed in various corners of the city, requiring a massive demand for analysis and computing resources, and each system model is independent of the other. With the help of AI agents based on 6G networks, previous systems such as emergency dispatch systems and intelligent transportation systems can be intelligently interconnected, generating more reasonable models and scheduling plans.

Metaverse. By interacting with the real world through computer-generated virtual environments, valuable, complex, and massive industry equipment such as aerospace, chip manufacturing, medical devices, transportation, and other industries are presented in virtual practice environments to provide an immersive interactive experience. XR and metaverse require much data transmission and processing, including real-time image, audio, and video streams. Therefore, the network needs to provide high bandwidth to support the transmission of these contents and ensure low latency to achieve real-time interaction and immersive experience. Through the 6G network and the AI agent's intelligent native capabilities, it can provide a larger transmission bandwidth and process data closer to where it is generated. With the help of AI agents, large-scale concurrent demands can be handled more reasonably without affecting the network, thereby reducing data processing latency and allocating computing resources more efficiently.

Challenging Issues for AaaS

Designing such an AI-native framework in 6G edge computing faces non-trivial challenges in the following four aspects.

1) Mutually Coupled Edge Resources. Both academia and industry have reached a consensus that the integration of sensing, and communication resources is the cornerstone of 6G networks [3], which creates a cyber-physical continuum. However, prior works on planning mainly focus on solving tasks within the cyber realm, such as arithmetic and common sense reasoning, while giving limited exploration of the physical world. Hence, how to achieve physical-instructed planning in AaaS with the intertwined edge resources of sensing and communication is a challenging problem.

2) Heterogeneous Edge Devices. In the 6G network, various networking entities such as the Central Units (CUs), Distributed Units (DUs), or even Radio Units (RUs) will be equipped with AI capabilities inherent to their hardware design, enabling them to operate as edge devices within the AaaS framework. These edge devices, which are designed for distinct networking roles, are heterogeneous in terms of capabilities and hardware architectures. For example, CUs are expected to facilitate high-end hardware even domain-specific accelerators (e.g., FPGA, GPU, NPU) while RUs only equipped with limited resources. Under this circumstance, how to support agents take full use of the hardware capabilities of heterogeneous edge devices is an unsolved problem.

3) Geo-Distributed Edge Deployments. Another key aspect of 6G is its global coverage, ranging from terrestrial, satellite, maritime, and even underwater communications scenarios. This wide-area coverage implies that AI agents will be geographically distributed under different scenarios. To make things worse, the generation and operation of agents in AaaS are managed by LLMs, which, due to their resource-intensive nature, cannot be deployed on all the edge devices in the 6G network. Therefore, how to schedule the agent execution among geo-distributed edge devices is a non-trivial challenge.

4) Evolvability of Edge Agents. In AaaS, AI agents provide services in both the data plane and control plane, which means they not only fulfill the edge computing tasks but also the network operation tasks. As an AI-native framework, the self-evolution ability of agents is important to cope with the data drift of different service scenarios [11]. Moreover, due to the complicated dependency of network parameters in the control plane, the ability to learn from deployment experiences is also important. Hence, how to facilitate the evolvability of edge agents after the initial generation is one of the key issues when designing AaaS.

Agent-as-a-Service: An AI-Native Framework

In this section, we first introduce the overview of the AaaS framework. Subsequently, we will introduce the user-space and kernel-space agents, as well as the AaaS infrastructure.

Overview of AaaS Framework

Fig. 1 showcases the proposed AI-native edge computing framework. AaaS regards the AI agent as a first-class citizen. We borrow the user space and kernel space concepts from the operating system and divide the agents in the AaaS framework into those two spaces. The user-space agents are responsible for serving application requirements, i.e., service intents, for both the data plane and control plane of 6G. Similar to the kernel-space components of the operating system, the kernel-space agents in AaaS manage the lifecycle of the user-space agents. Moreover, the AaaS framework also includes an infrastructure layer to provide basic execution environments such as agent runtime, deviceless orchestrater, and toolboxes for both data-plane and control-plane agents.

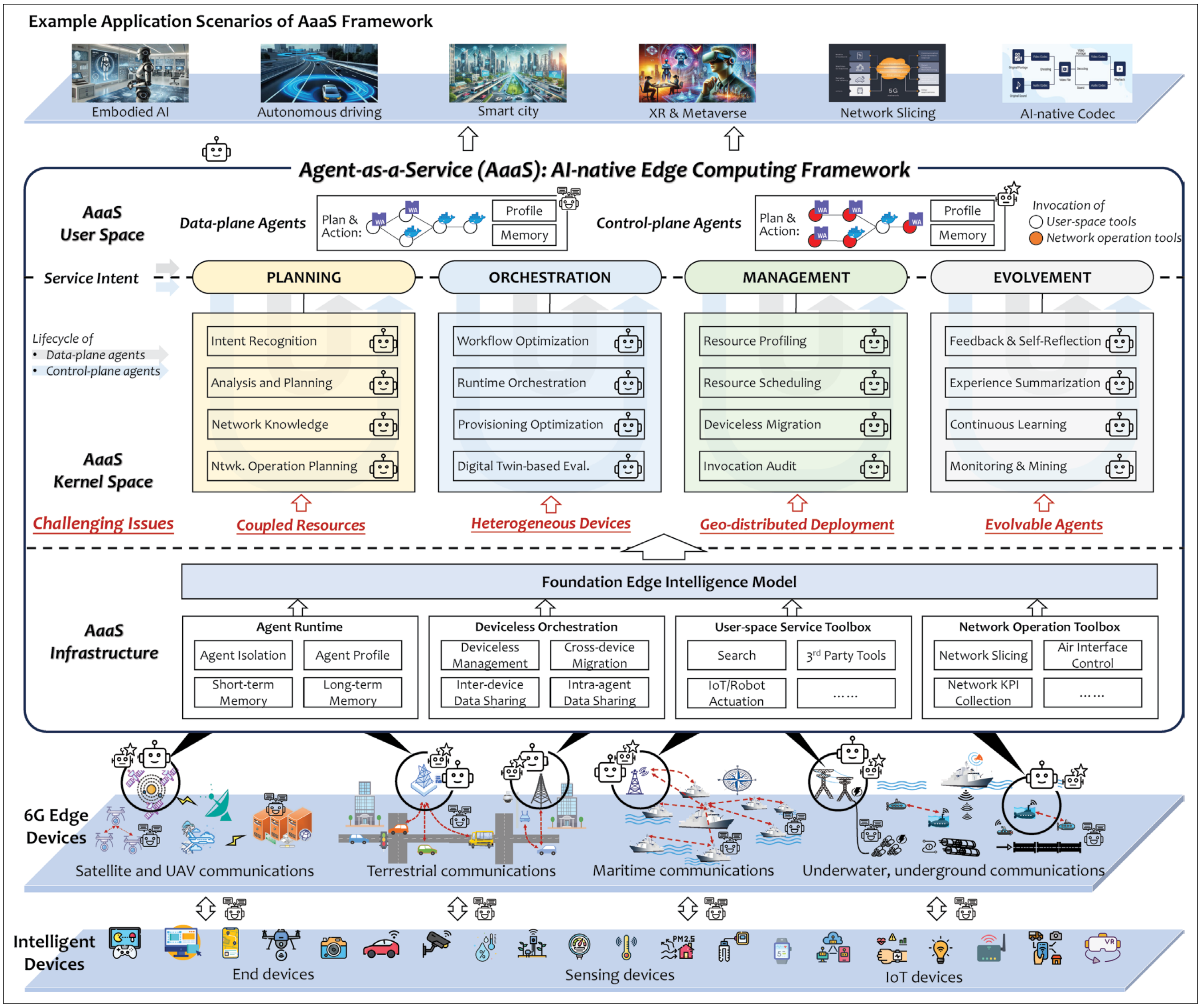

FIGURE 1. Overview of Agent-as-a-Service (AaaS), the proposed AI-native edge computing framework for 6G network. The component labeled with a robot icon denotes it is an AI agent.

AaaS User-Space Agents

To distinguish the functions of different agents, the AaaS framework divides agents into two categories: user-space agents and kernel-space agents.

The user-space agents are generated according to the service intents submitted by edge computing users or network operators. According to the intent, user-space agents can serve in the data-plane and control-plane of the 6G edge computing network. The former is for intents submitted by users with edge computing requirements such as image recognition and embodied robot controlling, which integrates sensing, communication, and computing tasks. The latter is for network operation and maintenance personnel to handle network management and network operations requests, such as monitoring the network status and configuring the network with network slicing. The main distinction between data-plane and control-plane agents is the different abilities to invoke tools during their actions. For network stability and operational security, only control-plane agents can invoke both the user-space service tools and the network operation tools.

Managing Agent Lifecycle in AaaS Kernel Space

The AaaS framework divides the agent lifecycle into the following four stages. When receiving a service intent, AaaS plans actions according to the intent and orchestrates the planned workflow. Subsequently, the AaaS framework deploys the agent, performs online resource management, and facilitates the continuous evolvement of agents. The lifecycle of control-plane and data-plane agents is different in AaaS because additional aspects should be considered to ensure the efficiency and integrity of control-plane agents. The distinction is denoted with arrows in different colors in Fig. 1 We now describe the detailed designs of each stage while taking the challenging issues mentioned in the section “Challenging Issues for AaaS” into consideration.

Planning. To address the challenges of mutually coupled edge resources, the AaaS framework introduces *resource-aware planning*.

In this approach, AaaS first analyzes the intent of the request, plans different ways of completing the request, and selects the plan according to its resource demands the available resources in edge devices. For the service intents, the AaaS framework constructs the process of completing this intent utilizing the Chain of Thoughts (CoT) or Tree of Thoughts (ToT) methods to break a complicated intent into steps. Utilizing CoT or TOT can also help to ensure logical consistency in multi-step reasoning tasks, thereby improving the reliability of generated plans. In addition, generating malicious networking operations may lead to severe consequences. Hence, AaaS utilizes Retrieval-Augmented Generation (RAG) to mitigate hallucinations in large language models and enhance agent security for networking operations.

Orchestration. Agents in this stage optimize the workflow generated by planning and then orchestrate it to executable instances. As edge devices are shared by multiple agents from various users, i.e., multi-tenancy, an isolation runtime for agents is a necessary design consideration. In cloud computing, the de facto runtime is the container. However, due to the heterogeneity in edge environments, the container is too heavy-weight for resource-constrained edge devices in terms of memory usage and cold-start time. Fortunately, an emerging lightweight bytecode-level isolation technique named WebAssembly [12] can execute on heterogeneous hardware with minimal runtime overhead. However, directly substituting containers with WebAssembly may not always be the optimal solution, primarily because of its limited support for bulk data processing [13]. Hence, during generation, a runtime orchestration agent dynamically selects the runtime considering the plan's affinity to heterogeneous devices, e.g., CPU, GPU, and NPU, on edge devices. Furthermore, the AaaS framework optimizes agent placement and resource provision aspects while being fully aware of the heterogeneous edge devices. Besides the executable generation of data-plane agents, the AaaS framework additionally supports control-plane agents with the ability to generate the network configurations and simulate its performance based on the network digital twin.

Management. In this stage, the AaaS framework completes the online deployment of agents and, more importantly, manages the resources scheduled for the agent during execution. The management agents first profile the resource usage of each user-space agent as well as dynamically schedule the network, compute, and storage resources to meet the resource requirements of agents and improve the overall performance of the system. Furthermore, the agents can be migrated among different edge devices during execution. However, edge devices in the 6G network are geo-distributed in a wide area, which poses challenges for resource management. To facilitate the dynamic yet seamless management over distributed edge devices, AaaS proposes a *deviceless approach*. The term "deviceless" originates from the serverless computing paradigm in the realm of cloud computing. It denotes that the instances of agents are automatically scaled according to the ingress requests. Once the agent is not requested for a short period, i.e., keep-alive time, it can be scaled to zero to save precious resources on edge devices. To ensure the integrity of tool usage, AaaS leverages several invocation audits in this step, including model interpretive checks and anomaly detection.

Evolvement. The evolvement stage monitors and handles errors, faults, and anomalies to ensure the stability and reliability of the AaaS framework. The changes in the external environment will affect the performance of the agent, such as the fluctuation of the request volume and the change of the network topology. To address the challenges of the evolvability of edge services, the AaaS framework employs continuous learning to adapt to new situations and requirements. In addition, for control-plane agents, the AaaS framework analyzes and models network operation patterns, traffic distribution, and other behaviors as network operation knowledge for further self-evolvable planning.

Through this four-stage lifecycle of agents, the AaaS framework can implement agent planning, orchestration, management, and evolvement to meet the needs of different intentions and automate the management of the entire lifecycle of the agent. Lifecycle management enables agents to provide services efficiently and reliably, and to adapt to changes in the environment and requirements.

Building Blocks of AaaS Infrastructure

The AaaS infrastructure is designed with four key components to support the diverse functionalities and operational needs of AI agents. This four-part division is based on the critical functional needs of AI agents and the foundational requirements for supporting both data-plane and control-plane operations efficiently.

The *Agent Runtime* serves as the execution environment for the agents, which is indispensable for the Agent-as-a-Service framework, directly supporting its main functionalities, including the profile, memory, planning, and action modules. The *Deviceless Orchestration* infrastructure supports the dynamic scaling and migration of AI agents across different edge devices, which is particularly crucial in geo-distributed deployments. This part of the infrastructure provides the fundamental abilities for the generation and optimization steps to invoke when facing the heterogeneous devices challenging issue in 6G edge computing.

There are also two toolboxes for the user-space and kernel-space agents to invoke. The *User-space Service Toolbox* serves as a flexible, invokable tool pool for user-space agents in both the data plane and control plane, such as searching, third-party integrations, and IoT/robot actuation. The *Network Operation Toolbox* offers a set of tools for control-plane agents to interact directly with the network infrastructure, enabling access to network status, management functions, and control capabilities. This component allows AI agents to obtain network-related data and execute network-specific

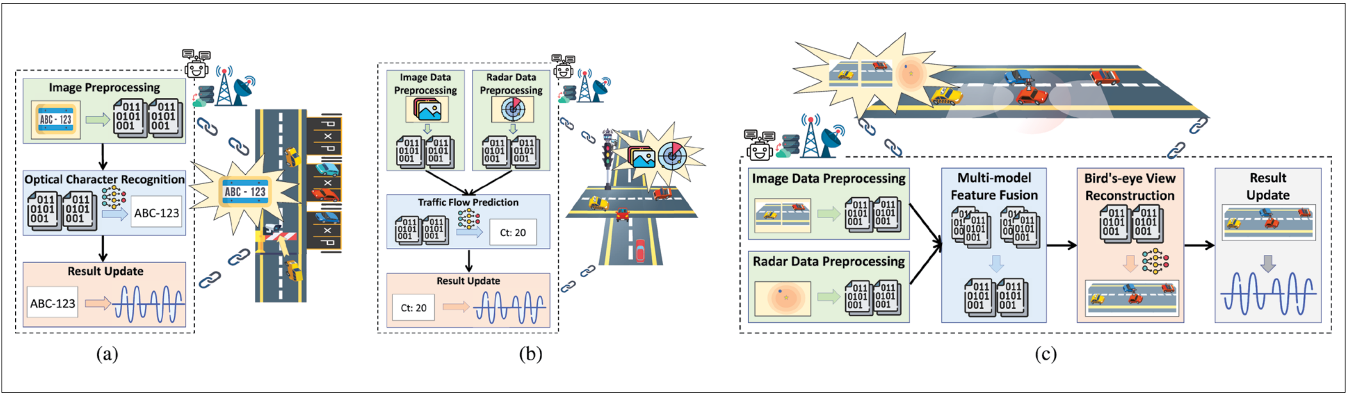

FIGURE 2. Illustration of the generated plan of three case studies in the autonomous driving scenario. a) #1: License plate recognition. b) #2: Traffic flow prediction. c) #3: Blind zone completion.

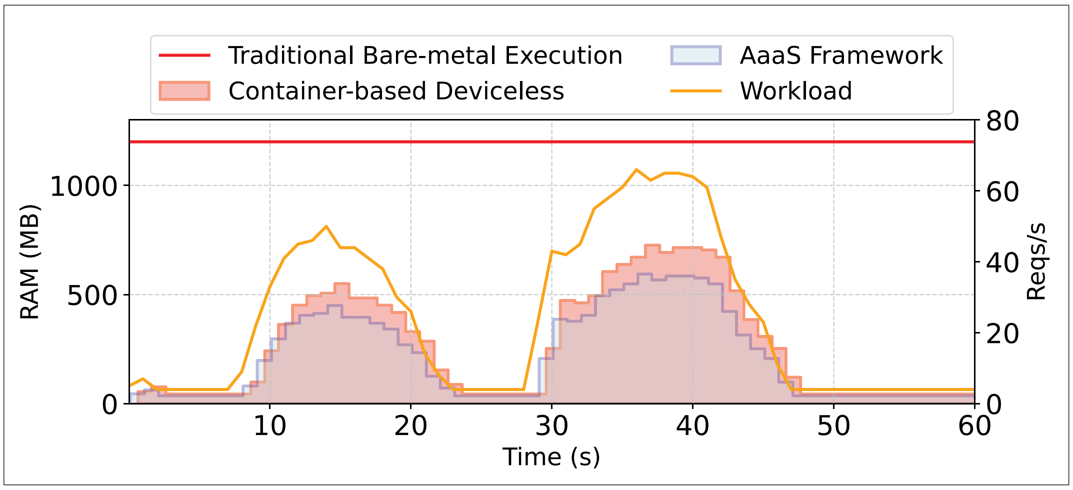

FIGURE 3. RAM consumption of different frameworks of the license plate recognition case.

operations, enhancing the robustness of network management.

Case Study

To showcase the effectiveness of the AaaS framework, we select three cases in the autonomous driving scenario for analysis and evaluation.

Experimentation Setup

We use three real-world cases to demonstrate how the AaaS framework works in the autonomous driving scenario.

The first case is *license plate recognition*. This case identifies and extracts vehicle license plate numbers from images, widely used in traffic management, parking management, and law enforcement. In autonomous driving, license plate recognition is crucial for collision avoidance and traffic monitoring.

The second case is *traffic flow prediction*, which uses historical traffic data and real-time observation to forecast future traffic conditions, such as vehicle flow and average speed. In autonomous driving, traffic flow prediction is crucial for route planning and driving safety. It helps the autonomous driving system to avoid congested roads in advance, optimize driving routes, and improve travel efficiency.

The third case is *blind zone completion*. It fuses the image from multiple cars and forms a panoramic picture of the intersection with the blind spots filled. It ensures the vehicle can fully perceive its surrounding environment during driving and avoid collisions and accidents.

To support the above cases, we build a proof-of-concept AaaS framework and evaluate both the agent planning results qualitatively and the execution efficiency quantitatively. We implement the planning step on top of state-of-the-art autonomous driving agent generation approaches [14], [15]. Moreover, we also integrate an edge computing toolbox tailored for autonomous driving scenarios. The toolbox consists of the preprocessing algorithms for image, video, and radar data as well as several deep learning models that are widely used in the autonomous driving scenario, such as optical character recognition and panoramic picture construction, for the generated agents to use.

Based on the generated plan, AaaS further implements the following steps of the four-stage lifecycle of agents. To evaluate the efficiency of our deviceless approach with multiple isolation technologies, i.e., container and WebAssembly, we also implement the traditional bare-metal execution without isolation, and the container-based

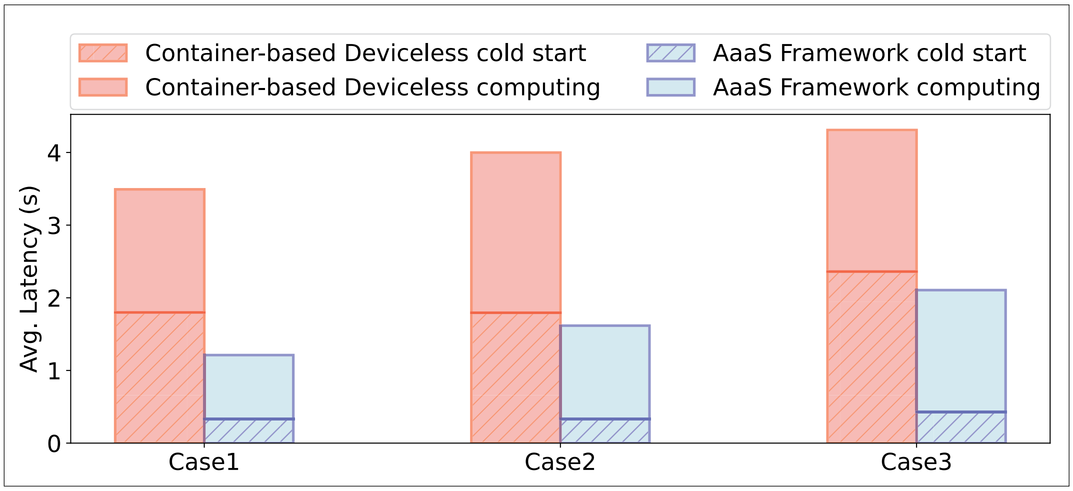

FIGURE 4. Average latency of different frameworks under different cases in autonomous driving.

AaaS framework paves the way for deeper integration of edge computing and network architecture and envisions a concrete instance of pervasive intelligence in 6G.

approach that is commonly used in cloud-based applications.

Performance Evaluation

Qualitative Analysis. First, we analyze the generated plans for each case study qualitatively. The planning results are demonstrated in Fig. 2.

Case #1: The generated plan indicates that the license plate recognition task is divided into three subtasks: image preprocessing, optical character recognition, and result retrieval. Following the deviceless architecture, each step is generated to an isolated instance, enhancing system elasticity and efficiency. We can see from this case that the generated plan of the AaaS framework can fulfill the service intent and the deviceless architecture increases the overall scalability.

Case #2 and #3: The traffic flow prediction task is divided into subtasks: data preprocessing (with different modalities processed in parallel), model prediction, and result return. Similarly, the blind zone completion task is planned with multiple sensor data preprocessing, feature fusion, and bird'seye view reconstruction models. In both case studies, AaaS enables parallel processing of different modalities and large-scale data, improving the real-time performance and accuracy of predictions during the optimization state. This ensures the generated plan can efficiently and safely perform service intents in complex environments.

Quantitative Results. We then evaluate the efficiency of the generated agents with the generation and optimization steps of AaaS. Our evaluation mainly focuses on two aspects: the RAM consumption and average invocation latency of agents in different cases of autonomous driving. We use the real-world traffic flow of Los Angeles County and the Bay Area for the task load requests.

The comparison of RAM usage is shown in Fig. 3. Due to space constraints, we only present the RAM usage comparison for Case #1. We can see from the figure that the AaaS framework can achieve the fastest response to load requests while requiring the least amount of RAM. This is because the AaaS framework selects the optimal runtime for each agent, leveraging the advantages of both WebAssembly and container. WebAssembly has a significant advantage over containers in terms of cold start, enabling rapid response to load requests. Moreover, most agents running on WebAssembly consume fewer computational resources since WebAssembly operations are closer to the native system layer. However, not all agents are suitable for WebAssembly. For instance, during the model inference stage in case 1, WebAssembly incurs higher computational costs compared to container due to the need for WebAssembly-based model inference to be implemented via the WASI interface.

As shown in Fig. 4, the AaaS framework achieves lower average latency across different case scenarios compared to other solutions. This is because we not only leverage the efficient cold start advantages of WebAssembly but also fully consider the operational efficiency of different agents in different runtimes. For example, in the three cases involving ML tasks, WebAssembly incurs higher computation time delays due to the need for computation based on external interfaces (e.g., wasi-nn). However, some simple ML tasks still benefit from WebAssembly's computational efficiency. Moreover, for most common computations, such as data preprocessing tasks illustrated in Fig. 2 or tasks that do not rely on external interfaces, WebAssembly typically achieves superior performance.

Conclusion and Future Research Directions

In this paper, we present Agent-as-a-Service, an AI-native edge computing framework for 6G. Under the promise of ubiquitous intelligence of the 6G network, the AaaS framework is centered on AI agents and manages the whole lifecycle of agents with AI. AaaS framework paves the way for deeper integration of edge computing and network architecture and envisions a concrete instance of pervasive intelligence in 6G. We also believe that the AaaS framework can be a basis for more research on edge intelligence for 6G.

We summarize the potential open research directions as follows.

Optimization of LLMs on Edge Devices. The whole AaaS framework is built upon the reasoning and planning capability of LLMs. However, in the mobile edge computing environment, especially under integrated communication and computation vision of 6G, LLMs also reside on the network devices. Resource demands of LLM inference may squeeze the optimization space of networking operations and interfere with communication tasks. Hence, it requires optimizing the LLMs in terms of memory usage and computation efficiency on edge devices.

Context-Aware of Agent Planning. Existing research on agent planning focuses on how to generate the right sequence of actions. In the real-world setup of AaaS, controlled planning that generates the orchestration of tools according to the system contexts such as available resources, service invocation patterns, and service level agreements (e.g., latency, throughput) of agents is an open question.

Security of Agent Actions. For the control-plan services powered by agents, it is vital to evaluate whether the generated plans and operations are secure or not. Although the AaaS framework leverages tool invocation audit during the operation stage for security enhancement, there is a vast space for further exploration and research in this area.

Acknowledgment

We thank all the reviewers for their insightful feedback to improve this paper. This work was supported in part by the National Natural Science Foundation of China under Grant 62302096, Grant 62272098, and Grant U24B20152; in part by the Natural Science Foundation of Jiangsu Province under Grant BK20230813; in part by the Central University Basic Research Fund of China under Grant 2242024K40019; and in part by the Zhishan Young Scholar Program of Southeast University under Grant 3209002402A2.

References

[1] Q. He et al., “Pyramid: Enabling hierarchical neural networks with edge computing,” in *Proc. ACM Web Conf.*, 2022, pp. 1860–1870.
[2] J. A. Ayala-Romero et al., “vrAIn: Deep learning based orchestration for computing and radio resources in vRANs,” *IEEE Trans. Mobile Comput.*, vol. 21, no. 7, pp. 2652–2670, Jul. 2022.
[3] C.-X. Wang et al., “On the road to 6G: Visions, requirements, key technologies, and testbeds,” *IEEE Commun. Surveys Tuts.*, vol. 25, no. 2, pp. 905–974, 2nd Quart. 2023.
[4] X. Shen et al., “Holistic network virtualization and pervasive network intelligence for 6G,” *IEEE Commun. Surveys Tuts.*, vol. 24, no. 1, pp. 1–30, 1st Quart., 2022.
[5] *Framework and Overall Objectives of the Future Development of IMT for 2030 and Beyond*, document ITU-R M.2160-0, 2023.
[6] W. Wu et al., “AI-native network slicing for 6G networks,” *IEEE Wireless Commun.*, vol. 29, no. 1, pp. 96–103, Feb. 2022.
[7] A. R. Hossain et al., “AI-native for 6G core network configuration,” *IEEE Netw. Lett.*, vol. 5, no. 4, pp. 255–259, Dec. 2023.
[8] J. Hoydis et al., “Toward a 6G AI-native air interface,” *IEEE Commun. Mag.*, vol. 59, no. 5, pp. 76–81, May 2021.
[9] Y. Yang et al., “TaskOriented 6G native-AI network architecture,” *IEEE Netw.*, vol. 38, no. 1, pp. 219–227, Oct. 2024.
[10] L. Wang et al., “A survey on large language model based autonomous agents,” *Frontiers Comput. Sci.*, vol. 18, no. 6, Dec. 2024, Art. no. 186345.
[11] Z. Li, M. Zhang, and Y. Zhu, “OAVS: Efficient online learning of streaming policies for drone-sourced live video analytics,” in *Proc. IEEE/ACM 32nd Int. Symp. Quality Service (IWQoS)*, Jun. 2024, pp. 1–10.
[12] A. Haas et al., “Bringing the web up to speed with WebAssembly,” in *Proc. 38th ACM SIGPLAN Conf. Program. Lang. Design Implement.*, Jun. 2017, pp. 185–200.
[13] A. Jangda et al., “Not so fast: Analyzing the performance of webassembly vs. native code,” in *Proc. USENIX ATC*, 2019, pp. 107–120.
[14] Z. Yang et al., “DoraemonGPT: Toward understanding dynamic scenes with large language models (exemplified as a video agent),” in *Proc. 41st Int. Conf. Mach. Learn. (ICML)*, 2024, pp. 1–11.
[15] J. Mao et al., “A language agent for autonomous driving,” in *Proc. 1st Int. Conf. Lang. Modeling (COLM)*, 2024, pp. 1–9.

Biographies

Borui Li (Member, IEEE) (libr@seu.edu.cn) received the Ph.D. degree in computer science from Zhejiang University, China, in 2022. He is currently an Assistant Professor with Southeast University. He has published research papers in prestigious conferences and journals, including NSDI, MobiCom, ATC, INFOCOM, IEEE Transactions on Computers, and TOSN. His research interests include the AI-native IoT, WebAssembly, edge computing, and deviceless computing. He is a member of ACM.

Tianen Liu (toliutianen@seu.edu.cn) received the B.S. degree from the School of Computer and Control Engineering, Yantai University. He is currently pursuing the Ph.D. degree with the School of Computer Science and Engineering, Southeast University. His research interests include autonomous driving, real-time scheduling, and edge computing.

Weilong Wang (wang_wl@seu.edu.cn) received the M.S. degree from the School of Computer and Control Engineering, Yantai University. He is currently pursuing the Ph.D. degree with the School of Computer Science and Engineering, Southeast University. His research interests include serverless computing, WebAssembly, and edge computing.

Chengqing Zhao (chengqingzhao@seu.edu.cn) received the M.S. degree from the School of Computer Science and Technology, Xi’an University of Posts and Telecommunications. He is currently pursuing the Ph.D. degree with Southeast University. His research interests include the Internet of Things, wireless communication protocols, and edge computing.

Shuai Wang (Senior Member, IEEE) (shuaiwang@seu.edu.cn) received the B.S. and M.S. degrees from the Huazhong University of Science and Technology, China, and the Ph.D. degree from the Department of Computer Science and Engineering, University of Minnesota, in 2017. He is currently a Professor with the School of Computer Science and Engineering, Southeast University. He has been the Head of the Computer Engineering Department since November 2022. His research interests include the wireless networks and sensors, cyber-physical systems, and data science.