Proceedings of the Thirty-Fifth International Joint Conference on Artificial Intelligence (IJCAI-26)
Survey Track

LLM-Based Agents on the Edge: A Survey of Privacy, Scalability, Heterogeneity, and Autonomy

Nikita Agrawal[1] and **Ruben Mayer**[1]
[1]University of Bayreuth
{Nikita.Agrawal, Ruben.Mayer}@uni-bayreuth.de

Abstract

Large language model (LLM)–based agents are increasingly being deployed beyond centralized cloud environments and toward the edge of the network, where they operate closer to data sources. This transition facilitates lower latency and enhances contextual awareness, privacy, and responsiveness, but it also introduces challenges that differ from traditional cloud-based agent deployments. This survey provides a systematic overview of LLM-based edge agents with a particular focus on four critical dimensions: privacy, scalability, heterogeneity, and autonomy. To facilitate structured analysis, we introduce a novel taxonomy along four axes: deployment, functional role, interaction topology, and adaptation pattern. Based on our taxonomy, we analyze the challenges LLM-based agents face on the edge and discuss design solutions that can help mitigate possible issues. We further analyze the degree to which existing LLM-based edge agent frameworks achieve privacy, scalability, heterogeneity, and autonomy.

1 Introduction

Large language model (LLM)-based agents have seen rapid growth as they have proven to be effective at multi-step planning, tool use, and reasoning. Recent works are moving these agents from cloud deployment to edge environments, closer to data sources such as smartphones, wearables, and IoT systems. This shift is motivated by the need for lower latency, faster responsiveness, and improved local context awareness. Studies such as [Jiang *et al.*, 2024; Chen *et al.*, 2025] demonstrate how LLM-based agents on the edge can autonomously interpret sensory inputs, coordinate actions, and make use of external tools or APIs in real time. However, edge environments are resource-constrained, less reliable, and often have access to sensitive data, introducing challenges that may not be prevalent in cloud environments.

When edge nodes process sensitive personal, sensor, or industrial data, deploying LLM-based agents at the edge raises significant privacy concerns. Edge environments usually involve many loosely connected devices communicating with one another, which may lead to data and prompt leakage during interactions among agents, especially in the presence of insecure communication channels, compromised nodes, or insufficient access control mechanisms [Zhang *et al.*, 2021]. Moreover, edge environments have constraints on compute, memory, energy, and network bandwidth, which make scaling challenging. Supporting many parallel agents or users across edge nodes can quickly exhaust local resources, especially when agents engage in multi-step reasoning or frequent communication [Hua *et al.*, 2023; Semerikov *et al.*, 2025]. Edge environments are inherently heterogeneous, as they support a wide range of devices with diverse hardware capabilities, varying network reliability, and heterogeneous data characteristics, adding additional challenges for coordination, optimization, and robustness [Shi *et al.*, 2016]. Edge agents are expected to operate with limited human supervision and unreliable connectivity to the cloud, making autonomy extremely important. However, increasing autonomy in LLM-based edge environments can lead to additional challenges in safety, control, and reliability [Cemri *et al.*, 2025; Liu *et al.*, 2024].

The complex interplay between edge computing and agentic AI motivates this survey. Existing surveys primarily concentrate on the deployment of LLMs either at the edge or in multi-agent systems, but not both simultaneously. For instance, studies presented by [Gill *et al.*, 2024; Semerikov *et al.*, 2025] emphasize the deployment of LLMs in resource-constrained edge or mobile environments. These works focus on aspects such as model efficiency, compression, and system optimizations while only briefly addressing the challenges of managing diverse agents, their autonomous behaviors, and the risks associated with deploying them in edge settings. Other studies provide only partial analyzes of issues related to privacy, scalability, heterogeneity, and autonomy. [Kokkonen *et al.*, 2022] focus primarily on high-level architectural strategies to enhance local autonomy and handle heterogeneous resources while maintaining data privacy in device-edge-cloud architectures but offer only a partial outlook on scalability and coordination challenges in fully autonomous edge systems. Meanwhile, [Jiang *et al.*, 2024] analyzes LLM-enhanced multi-agent systems mainly from a communication and coordination perspective, discussing little about autonomy and heterogeneity while leaving privacy risks and scalability challenges largely unexplored. In this

survey, we present a taxonomy of LLM-based edge agents structured around four axes: deployment, functional roles, interaction topology, and adaptation pattern. We subsequently examine agent subtypes across these axes, focusing on privacy, scalability, heterogeneity, and autonomy, and discuss associated challenges and mitigation strategies. This closes an important research gap and can serve as a basis for further research on agentic AI in edge computing.

2 Edge Computing and LLM Agents

Fundamentals of edge computing. Edge computing is a distributed computing paradigm that shifts computation and data storage from centralized cloud data centers closer to data sources. Processing data locally on devices or nearby edge servers drastically reduces round-time latency and bandwidth usage compared to cloud-only processing [Chen *et al.*, 2025]. This makes edge computing favorable for time-sensitive applications, such as real-time analytics, real-time response, and autonomous control. Generally, an edge computing system employs a multi-tier architecture, which comprises end devices, edge nodes, and central cloud data centers. Partitioning the computation across these tiers helps optimize performance, cost, and privacy [Shen *et al.*, 2025]. By limiting the transmission of raw data to the cloud, edge computing reduces exposure to data leakage, supports compliance with data protection regulations such as GDPR [European Union, 2016], and enables greater control over sensitive and personal information [Shi *et al.*, 2016].

From traditional edge ML to LLM-based agents. Traditional edge AI applications use small-scale models trained for specific tasks, such as a convolutional neural network (CNN) trained for vision or reinforcement learning (RL) policies for control [Gill *et al.*, 2024]. These models are optimized to perform in a constrained environment on edge devices. However, they are usually tailored to a predetermined function and have a limited ability to generalize beyond the distribution of training data or to modify their trained behavior at runtime [Liu *et al.*, 2024]. In contrast, LLM-based agents act as programmable, natural-language controlled systems characterized by planning, tool use, and memory [Jiang *et al.*, 2024]. They employ reasoning techniques such as Chain-of-Thought (CoT) to break down goals into small tasks and have the ability to interact with the outside world using external APIs through systematic tool calls [Wu *et al.*, 2024]. Moving from classical edge AI to LLM-based agents fundamentally changes the computational load on the network edge from static tasks such as pattern recognition to goal-oriented planning and decision-making by reasoning and coordination. Recent work explores LLM-based agents on the edge, either by using LLMs as high-level semantic controllers over traditional edge models or by optimizing sub-billion parameter LLMs to run directly on devices [Liu *et al.*, 2024].

In this survey, we assume that our LLM-based agents have an interface with local sensors or devices and obtain logs, events, and sensor data from them. We also assume that they have access to certain tools and memory systems. They may also have knowledge of a policy for autonomous reasoning and action.

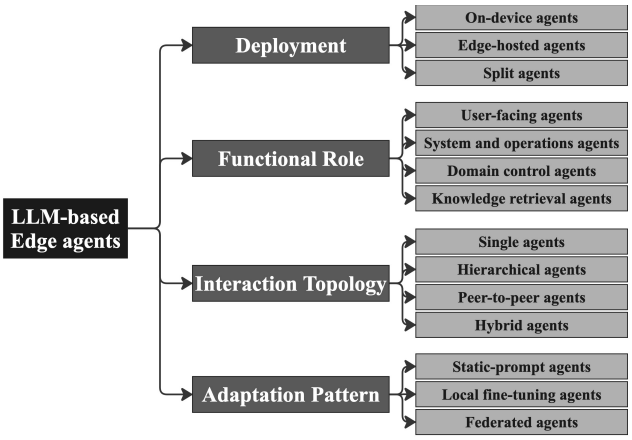

Figure 1: Our Taxonomy of LLM-based edge agents

3 Taxonomy of LLM-Based Edge Agents

LLM agents deployed on edge infrastructures exhibit specific architectural, system-level, functional, and learning properties. We propose a classification scheme along four axes: deployment, functional role, interaction topology, and adaptation pattern (cf. Figure 1). These axes cover the crucial design choices of LLM-based edge agents. We use the taxonomy to structure the discussion of edge-specific challenges of agentic systems.

Deployment. On the deployment axis, we consider where the LLM computation takes place. We can categorize LLM agents into the following deployment categories: On-device, Edge-hosted, and Split agents.

On-device agents run entirely on devices such as mobile phones, wearables, cars and robots. They make use of quantized or distilled LLMs along with limited tool use capabilities [Liu *et al.*, 2024]. These agents excel at privacy and latency, as computation takes place on the device, which implies that sensitive data do not need to be shared. The agent typically uses a smaller context window and needs to manage the KV-cache size to avoid running out of memory. Although these constraints reduce the quality of generated output when compared to larger models deployed on servers, they enable fully offline and privacy-preserving agent deployment.

Edge-hosted agents run LLM computations on edge servers or gateways that are computationally more powerful than local devices. They aggregate multiple requests coming from nearby devices. Edge-hosted agents can host larger LLMs and access more tools than local devices. Compared to cloud servers, edge servers have better control over data and lower latency. However, while edge systems may include a small number of relatively powerful servers, such as with server-grade GPUs, these resources are limited compared to the cloud, making it difficult for them to scale. Furthermore, heterogeneity in a distributed edge environment increases the management complexity [Hua *et al.*, 2023].

Split agents distribute computation across multiple tiers. Typically, a powerful base model is deployed in the cloud, and multiple edge nodes deploy lightweight and specialized components of larger models, such as LoRA modules or adapters. These help encode sensitive content and improve

personalization while running the heavy computation in the cloud [Wang *et al.*, 2023]. However, split agents might suffer from higher latency and are prone to network disruptions or cloud failures. This may degrade the responsiveness and robustness of the agentic system. Adding additional tiers provides greater computational power but increases the surface of privacy and security attacks [Shen *et al.*, 2025; Shi *et al.*, 2016].

Functional role. The functional role represents what the agent is responsible for in the system.

User-facing agents have a natural-language interface and are designed for direct interaction with end users. They translate natural-language prompts into actions such as API calls or database queries [Patil *et al.*, 2024] and can also be personalized through stored user preferences [Chen *et al.*, 2025]. Such agents provide low-latency access to local context, but they can also become a single point of failure for user workflows. As they directly interact with users, this can cause privacy concerns if sensitive data is processed [Shen *et al.*, 2025; Gill *et al.*, 2024].

System and operations agents do not directly interact with users. Instead, they manage edge infrastructure by leveraging LLMs and advanced techniques such as deep reinforcement learning to analyze the system state and execute complex management tasks. To manage the high volume of operational data, such as system metrics and logs, they use special memory architectures or context-aware compressors [Shen *et al.*, 2025]. Based on the monitored data, the agents decide how to allocate resources, adjust configurations, and invoke management APIs to meet service expectations. This enhances the manageability and responsiveness of distributed edge deployments by shifting system scaling from reactive, threshold-based to proactive, predictive orchestration capable of anticipating workload surges or hardware fluctuations [Mwangi *et al.*, 2025]. However, system agents when deployed on the edge might only have a partial or stale view of the overall system, which can lead to incorrect decisions that cause misconfigurations. Recent work on agents in edge network environments shows that even slight variations in an agent's logic can significantly degrade system performance, especially when agents are not strongly aligned with the operational stability goals [Salama *et al.*, 2025].

Domain control agents serve as high-level decision makers by translating abstract reasoning into concrete actions based on real-time sensor feedback. By focusing on a particular domain, these agents exploit domain-specific knowledge to improve performance and reliability, e.g., by analyzing digital twin data to coordinate system capabilities and functions [Xia *et al.*, 2023]. They perform actions such as controlling industrial equipment over a factory gateway or assessing security events at an enterprise edge firewall [Hasan *et al.*, 2024]. When these agents are located close to the domain data and the data sources, they can make low-latency and context-aware decisions that might be difficult to attain when relying solely on cloud-based processing [Mao *et al.*, 2024]. However, domain control agents also face challenges, such as bugs, hallucinations, or misalignment of objectives. To mitigate these risks, agents can include safety filters based on formal methods or models that take uncertainty into consideration when making decisions [Ding *et al.*, 2025; Ames *et al.*, 2017]. These measures ensure that agents' decisions remain within defined safety bounds, protecting physical constraints while preserving domain-level reasoning.

Knowledge retrieval agents index local data sources and provide retrieval-augmented generation as a shared service for other edge agents and devices. This architecture can be observed in frameworks such as AdaRAG, where each edge node maintains a local knowledge base for retrieval [Ouyang *et al.*, 2025]. Constraining the LLM output to local data helps improve the quality of the response and reduces the need for communication with the cloud. However, deploying retrieval capabilities on the edge introduces challenges related to limited storage, indexing overhead, and data consistency. It also poses the risk of agents using outdated or incorrect information to make critical decisions due to poor data indexing, potentially endangering critical systems.

Interaction topology. The interaction topology describes how agents and other components, such as tools and sensors, are organized. They enable a balance between local autonomy and global coordination.

The *single-agent* pattern consists of one LLM agent per device or per edge node. This agent is responsible for all reasoning, tool calling, and decision making. This agent pattern reduces communication overhead and ensures privacy, as data remains local, but at the cost of generalization and performance bottlenecks. Prior work shows that single-agent systems using smaller models deployed at the edge struggle with complex tool use and often fail to reliably plan multi-step API calls or recover from intermediate errors without external feedback [Shen *et al.*, 2024]. As a result, a single-agent system may not notice its own mistakes, motivating the use of additional agents that can review and correct its actions.

The *hierarchical-agent* pattern arranges multiple agents into levels such as device, edge, and cloud to manage task granularity. Device-level agents focus on immediate sensing and reactions, edge-level agents facilitate coordination among agents on the devices, and cloud-level agents handle high-level global planning. Current device-edge-cloud frameworks that distribute control to maintain low-latency responses while leveraging global control already instantiate multi-level hierarchical pattern agent systems [Kokkonen *et al.*, 2022]. However, hierarchical systems can suffer from misalignment and specification drift, where errors introduced by high-level planners propagate downward through the system. If a high-level planner produces incorrect task decomposition, the overall goal might fail even if low-level agents correctly execute their assigned subtasks. This problem is aggravated when context is abstracted or partially lost as information flows across hierarchical levels [Cemri *et al.*, 2025].

The *peer-to-peer multi-agent* pattern consists of multiple loosely coupled agents that assume the roles of specialized planners, retrievers, executors, and critics. These agents collaborate with each other through natural-language or structured messages. This decentralized setup enables agents to exchange perspectives, refine solutions, and collectively solve complex tasks more effectively than a single agent alone.

Although the peer-to-peer agent pattern offers flexibility, it also presents challenges, such as aligning agents with one another and handling the communication overhead. Extended multi-agent interactions can drift from the original task, reinforce incorrect reasoning, or allow a single agent to influence global actions without any centralized control. This can lead to errors and reduce reliability in peer-to-peer agent systems [Chan *et al.*, 2024].

The *hybrid multi-agent* pattern combines centralized orchestration with loosely coupled device-level peer agents. A coordinator agent at the edge assigns tasks, oversees safety, and resolves conflicts among device-level agents. The device-level agents are mostly semi-autonomous, handling local reasoning and execution. Adding a coordinator helps overcome the shortcomings of peer-to-peer agents by improving system reliability. However, it can be difficult to maintain synchronization between the coordinator and the local agents. If the coordinator takes too long to respond, it can slow down or completely block the local agents, resulting in interference with time-critical local actions [Cemri *et al.*, 2025].

Adaptation pattern. Adaptation patterns address the evolution of the agents' behavior over time in the system.

Static-prompt agents use frozen model parameters along with curated prompts to control behavior. Any change in behavior is made through prompts or tool configuration, without modifying the model itself. Since these agents do not update any parameters on the device, they are lightweight, predictable, and easy to deploy. However, they have limitations, such as declining performance on complex tasks and struggling to adapt to any new or evolving information, as they are restricted by the model's fixed context window. Any modification of the prompts can lead to inconsistent agent behavior, making them less reliable [Cao *et al.*, 2024].

Local fine-tuning agents use a shared base model while allowing individual edge devices to personalize behavior through parameter-efficient fine-tuning, such as LoRA, adapters, or prefix tuning. This encourages personalization or domain alignment without retraining the complete model and improves privacy by keeping data on the device [Wang *et al.*, 2023]. However, local fine-tuning can lead to the model focusing too much on new data and forgetting previously learned general knowledge.

Federated agents participate in distributed learning protocols. Each edge node in the system trains the local adapters or policies and periodically aggregates them. Such patterns provide a way for agents to scale for global improvement while preserving the privacy of sensitive data. However, federated agents face challenges related to heterogeneity, such as delays in training due to slower devices and bias in the aggregated model because of differences in local data [Hu *et al.*, 2024].

4 Design Considerations: Privacy, Scalability, Heterogeneity, Autonomy

LLM-based edge agents operate in resource constrained environments with sensitive data and need to make decisions autonomously. Although edge deployment promises low latency and improved data locality, it also requires design trade-offs to overcome challenges concerning privacy, scalability, heterogeneity, and autonomy. A systematic study is required to understand how different classes of LLM-based edge agents can be deployed safely, efficiently, and robustly. In each of the subsections below, we begin by reviewing the common strategies and concerns related to the given design consideration, and then provide an in-depth analysis of how various agent subtypes, distinguished by their deployment context, functional role, interaction topology, and adaptation strategy, address or are impacted by the issues associated with that consideration.

4.1 Privacy

Privacy is one of the primary goals of moving computation-heavy LLMs towards the edge. However, deploying LLM-based agents on the edge introduces many privacy threats, depending on where the data is stored and processed. Sensitive data can leak during data exchanges across local, edge, and cloud agents. Edge agents must comply with regulatory constraints such as GDPR [European Union, 2016] while enhancing their contextual awareness. Thus, privacy-enhancing technologies are critical for improving privacy. In this section, we analyze the data risks associated with different taxonomy axes and discuss potential design solutions.

Deployment: On-device agents maximize privacy by processing data locally on the device. Similarly, deploying agents on organizational edge servers keeps data within trusted boundaries, where personal identifiers can be stripped out or generalized using techniques such as k-anonymity before any information is transmitted to the cloud [Wei *et al.*, 2018]. Privacy in split-agent architectures can be improved by carefully placing the initial or personalization layers on the edge while offloading computationally intensive attention blocks of the LLM to the cloud, with only intermediate activations exchanged between the two [Wang *et al.*, 2023]. This reduces direct data leakage but risks inference attacks on exchanged activations. Safeguards such as encryption [Karthikeyan *et al.*, 2025] and communication over secure channels help mitigate the risk.

Functional: Agents with different functional roles handle specific types of privacy-sensitive data, such as users' personal data, conversation logs, system metrics, and domain-specific data. To preserve privacy, the agents should strip data of private identifiers and generalize it using k-anonymity [Wei *et al.*, 2018] or differential privacy [Dwork, 2006]. Any external communication of the agents, such as external APIs or tool calls, should be conducted through secure communication channels such as HTTP/TLS. Additional security measures can involve using enclaves such as trusted execution environments to shield the called tool or service [Karthikeyan *et al.*, 2025].

Interaction: In multi-agent systems, such as hierarchical agents, peer-to-peer agents, and hybrid agents, interaction patterns influence privacy. Any data exchange among such agents must be carefully governed. Each agent in multi-agent systems must receive only the minimal information necessary for the task. A high-level coordinator, if present, must remove any sensitive details and encrypt the data before communicating it to other agents [Karthikeyan *et al.*, 2025]. Central

coordinators must be secured, as they can be a single point of failure if compromised. In peer-to-peer agents, when agents communicate directly with each other, mutual authentication can mitigate the risk of eavesdropping [Du *et al.*, 2024]. Any kind of agentic system, even a single-agent system that stores data or models centrally, may be vulnerable to privacy attacks, such as membership inference, data reconstruction, and inadvertent cross-context leaking [Fu *et al.*, 2024].

Adaptation: Beyond privacy attacks on data exchanged by agents, there are potential privacy leaks in learning agents. Static-prompt agents that do not learn are mostly private, as they do not collect data beyond prompt updates. Local fine-tuning agents and federated agents attempt to maintain privacy by not sending any raw training data to the cloud. However, the trained model may still encode sensitive details that could be extracted. Differential privacy can prevent the agent from memorizing any individual record [Dwork, 2006]. In federated agents, the collaborative training process needs to be further secured, e.g., by secure aggregation or homomorphic encryption [So *et al.*, 2023; Zhang *et al.*, 2021].

4.2 Scalability

The scaling of LLM agents across edge nodes and devices concerns computational scalability, management scalability, and network scalability. Computational scalability refers to the ability to scale raw processing capacity. This is an important consideration, as slow agents can slow down the entire workflow [Gill *et al.*, 2024]. Network scalability concerns communication overhead and data movement, which become particularly important when agents frequently communicate or when thousands of devices concurrently interact with shared edge or cloud agents. Management scalability concerns the difficulty of deploying, updating, coordinating, and maintaining thousands of devices and agents as the system grows [Kokkonen *et al.*, 2022]. It must also account for resource and energy constraints at the edge, which indirectly impose cost considerations on large-scale deployments. Below, we examine how each type of agent architecture and strategy impacts scalability in terms of compute, communication, and manageability.

Deployment: On-device agents scale horizontally trivially, as each device hosts its own local LLM at the cost of maintaining many isolated instances [Liu *et al.*, 2024]. Edge-hosted agents can serve multiple concurrent sessions by scaling the edge node capacity. Their performance can be improved by load balancing across multiple edge-servers and by model sharing [Shi *et al.*, 2016]. Managing edge-hosted agents is easier compared to on-device agents, as updates target a limited number of servers rather than thousands of devices. Split agents enable load distribution across tiers, allowing them to support larger models and higher concurrency, leading to better LLM performance [Semerikov *et al.*, 2025]. They enable a cluster of edge nodes to provide highly-capable agents to many users instead of repeatedly hosting the whole model per edge node. However, continuous communication among multiple nodes can become a latency bottleneck. Mitigation techniques such as caching and batching requests can help. Thus, split-agent scalability requires robust connectivity and often involves multiple cloud endpoints to avoid single points of failure [Zhang *et al.*, 2021].

Functional: User-facing agents can be designed as on-device agents, edge-hosted agents, or split agents, and they can be scaled accordingly. Since they store user logs and conversational history, strategies such as summarizing context and caching common responses and prompts help enhance responsiveness [Xu *et al.*, 2024]. System and operations, domain control, and knowledge retrieval agents scale by partitioning responsibilities across multiple hierarchical agents, such that each agent oversees a subset of devices, domains, or functions. Coordination overhead can be reduced if each agent handles its own tasks without constant communication with other agents. Knowledge retrieval agents need to index and search through more information as data and queries grow, which can be challenging on limited hardware. They can be scaled by using data management techniques such as sharding, hierarchical indexing, topic-based partitioning, and caching frequent queries [Ouyang *et al.*, 2025].

Interaction: The single-agent pattern scales poorly, as one agent can provide only limited processing, making replication across devices or edge nodes essential. As single-agents struggle with complex tasks and multi-step planning, it may become necessary to switch to a multi-agent system [Shen *et al.*, 2024]. Hierarchical multi-agent systems scale efficiently as they partition responsibilities among device, edge, and cloud level agents, allowing complex workloads to be handled without overloading any single agent. However, communication overhead must be controlled, e.g. by exchanging only aggregated information [Zhang *et al.*, 2021]. Although the peer-to-peer multi-agent pattern offers the flexibility to scale, it increases communication overhead, potentially leading to network bottlenecks and high latency. Using a hierarchy or limiting the number of peers with whom each agent can communicate helps reducing communication overhead. In the hybrid-agent pattern, the central coordinator reduces the uncontrolled communication of the peer-to-peer network by directing messages and coordination through itself. However, the coordinator can become a bottleneck if it is unable to handle a growing number of agents. This can be addressed by introducing multiple coordinators or hierarchical layers to distribute the coordination load.

Adaptation: Static-prompt agents ease scalability because they are lightweight, do not require training, and are identical across deployments, making them easy to deploy on multiple devices and cache frequent queries to reduce latency. However, any changes to the agent require redeploying all instances. Local fine-tuning agents scale learning using standard parallelization techniques. However, they are difficult to coordinate, as each agent can drift in its own direction and cause a loss of global knowledge [Cemri *et al.*, 2025]. One solution to mitigate this issue is to periodically aggregate and synchronize the models, converting local fine-tuning agents into federated agents. Federated agents scale both data volume and computation by leveraging large-scale distributed training. However, they suffer from device heterogeneity and communication overhead, which can be mitigated by limiting the number of agents participating in each training round, compressing updates, and using parameter-

efficient fine-tuning techniques and asynchronous aggregation [Woisetschläger *et al.*, 2024].

4.3 Heterogeneity

Heterogeneity in LLM edge agents arises because devices, networks, workloads, and software stacks differ widely across the edge and cloud. This affects how agents are deployed, how they access tools and data, and how they coordinate. Managing heterogeneity is less about supporting multiple models and more about providing abstractions and mechanisms that allow many different configurations to behave coherently and safely.

Deployment: On-device agents adapt to smaller devices by scaling down the LLMs using quantization or smaller models to fit the device capabilities. This means that not all on-device agents will be identical, and some on-device agents may perform better and offer more functionalities than others [Xu *et al.*, 2024]. Edge-hosted agents operate on heterogeneous edge servers with varying compute and network conditions. They adapt by loading different model sizes, using caching and buffering to handle diverse client connections, and continuing local operations when cloud connectivity is limited [Hua *et al.*, 2023]. Split agents manage heterogeneity by partitioning model components across tiers, deploying a shared base model on edge or cloud infrastructure while keeping lightweight, personalized adapters on devices. However, they must tolerate heterogeneous network conditions and ensure compatibility between differently sized or personalized model components [Semerikov *et al.*, 2025].

Functional: User-facing agents face heterogeneity from diverse users, devices, input modalities, and tasks. They adapt through modular designs that enable device-aware optimizations, multi-modal processing, and personalized behavior. Local or federated learning further helps accommodate heterogeneous user preferences and data distributions [Zhang *et al.*, 2021]. System agents are designed to manage infrastructure heterogeneity across devices, operating systems, and networks by abstracting differences using standardized interfaces, asynchronous control, and hierarchical deployment of local and global agents. Domain control agents face heterogeneity in sensors, actuators, data formats, and operational rules within a domain, which they manage using domain models, standardized schemas, and modular adapters to normalize variability into coherent control decisions [Xia *et al.*, 2023]. Knowledge retrieval agents manage heterogeneity in data types, locations, and formats by embedding heterogeneous data into shared representations or by federating queries across multiple local sources. Standardized query interfaces allow other agents to access diverse knowledge consistently [Ouyang *et al.*, 2025].

Interaction: Single-agent systems manage heterogeneity by isolation: each agent accommodates only its local device and context. This simplifies their design but hinders the utilization of system-wide resources [Gill *et al.*, 2024]. Hierarchical multi-agent systems tackle heterogeneity by allocating tasks among devices, edge, and cloud layers according to their capabilities, with lower levels handling local data and higher levels functioning on aggregated views. Peer-to-peer agents deal with heterogeneity via self-organization techniques, including capability discovery, negotiation, and redundancy. Hybrid systems leverage heterogeneity through a central coordinator with global awareness of agents and network conditions to allocate tasks appropriately, thus enhancing resource utilization, but at the expense of the coordinator's availability and accuracy.

Adaptation: Static-prompt agents handle heterogeneity poorly because they cannot adapt dynamically to different devices, tasks, or contexts without prompt engineering. This limits their flexibility in dynamic environments [Cao *et al.*, 2024]. Local fine-tuning agents adapt models to local data and context to address heterogeneity, improving performance under non-IID data [McMahan *et al.*, 2017]. Federated agents designed to learn from heterogeneous data and devices allow clients with different capabilities to contribute to the global model by using weighted updates and partial participation. This enables collective learning across heterogeneity while maintaining a shared model [Lai *et al.*, 2022].

4.4 Autonomy

Autonomy concerns the degree to which LLM edge agents can perceive, decide, and act without direct human supervision. At the edge, autonomy is essential for responsiveness and robustness under intermittent connectivity.

Deployment: On-device agents operate with high autonomy for simple tasks, allowing them to function during network disruptions. However, for complex or high-risk tasks, they require external support and network connectivity [Liu *et al.*, 2024]. Edge-hosted agents support varying forms of autonomy, such as adaptive tool selection, performance optimization, anomaly response, and self-healing, commonly modeled as monitor–analyze–plan–execute feedback loops over shared knowledge. Split agents offer limited local autonomy by executing lightweight adapters at the edge, while relying on the centralized model for complex computation, making their autonomy dependent on network connectivity and coordination [Wang *et al.*, 2023].

Functional: User-facing agents have limited autonomy, usually focused on selecting tools and executing user instructions and strategies without setting independent goals. System and operations agents require higher autonomy to manage complex infrastructure at scale and perform monitoring, optimization, and recovery without human intervention. However, they are constrained by safety rules, hierarchical oversight, and offline validation to prevent failures. Although domain control agents require strong autonomy to make fast, continuous decisions in sensitive environments, their actions are restricted due to potential real-world impact, safety controllers, formal constraints, and regulatory limits. Knowledge retrieval agents are autonomous in maintaining, updating, and querying local knowledge sources [Ouyang *et al.*, 2025]. They act autonomously in maintaining, querying, and optimizing local knowledge sources to improve responsiveness and reduce cloud dependence, while constraining autonomy to avoid hallucinations or misuse of incomplete information.

Interaction: Single-agent systems have full autonomy for enabling rapid decision-making but require fixed rules, as the absence of peer verification makes them fragile [Cemri

Framework	Taxonomy	Privacy	Scalability	Heterogeneity	Autonomy
AgentsCoDriver	On-device, domain control, peer-to-peer multi-agent, static-prompt agents	–	–	–	✓
MobileLLM	On-device, user-facing, single, static-prompt agents	✓	✓	✓	✓
O-RAN	Edge-hosted, system and operations, hierarchical, local fine-tuning agents	–	✓	–	✓
Industry 5.0	Edge-hosted, system and operations, hybrid multi-agent, static-prompt agents	–	✓	–	✓
Agentyxcrypt	Edge-hosted, system and operations, peer-to-peer multi-agent, static-prompt agents	✓	–	✓	–

Table 1: Capabilities of existing LLM-based edge agent frameworks with respect to Privacy, Scalability, Heterogeneity, and Autonomy.

et al., 2025]. Hierarchical systems allow lower-level agents to react independently to local conditions while higher-level agents enforce global policies. This enables rapid local reactions while maintaining system-wide safety and governance. Peer-to-peer agents exhibit decentralized autonomy that maximizes flexibility and fault tolerance but risks coordination failures without strong protocols. Shared goals and conflict-resolution mechanisms are essential for coordination. Hybrid systems grant peers autonomy to execute their tasks while coordinators manage global constraints, reducing conflicts [Jiang *et al.*, 2024].

Adaptation: Static-prompt agents exhibit limited autonomy, as they cannot perform tasks beyond their predefined prompts [Cao *et al.*, 2024]. Local fine-tuning agents have controlled autonomy, as they can update their models based on local data but are restricted by constraints, such as validation checks and restricted updates, to avoid model drift or overfitting. Federated agents have high autonomy as they learn together without sharing data [McMahan *et al.*, 2017].

5 Analysis of Existing Frameworks

For practical adoption, frameworks for LLM-based edge AI agents must address privacy, scalability, heterogeneity, and autonomy. We analyze recent frameworks to determine their progress in addressing these concerns, along with the taxonomy they support (Table 1).

Recent frameworks illustrate different tradeoffs between privacy, scalability, heterogeneity, and autonomy. AgentsCoDriver [Hu *et al.*, 2024] has high autonomy as it enables collaborative autonomous driving by allowing vehicles to reason, communicate, and learn jointly at the edge. However, they do not implement any privacy mechanisms and assume that vehicles have similar resources. MobileLLM [Liu *et al.*, 2024] optimizes a sub-billion parameter language model into different sizes for privacy-preserving, autonomous on-device agents that can be scaled to devices with varying capabilities but require retraining for any change in task. O-RAN [Salama *et al.*, 2025] integrates AI-driven traffic prediction and neural architecture search into the O-RAN architecture to autonomously optimize radio access networks with minimal human oversight. It is highly scalable, as decision-making is distributed across edge nodes. However, it lacks privacy and heterogeneity, as it collects and analyzes network traffic data in a centralized control component and assumes that network equipment and data formats across the radio access network are mostly similar. The Industry 5.0 framework [Martinez-Gil *et al.*, 2025] offers a multi-agent system for rapid deployment of edge AI solutions in factories. It is highly autonomous, as agents can independently make decisions, such as monitoring equipment and controlling processes. It can also be easily scaled by deploying agents across multiple factory sites or machines. However, it does not concentrate on privacy and assumes that most machines follow similar standards, meaning it can struggle with heterogeneity if deployed in environments with varying hardware, data formats, and network conditions. Agentyxcrypt [Karthikeyan *et al.*, 2025] introduces a framework that preserves the privacy of sensitive data exchanged among agents by encrypting multi-tier communication and secure computation, making it a good choice for heterogeneous systems with different tasks, data types, and environments. However, it is not autonomous or scalable, as fixed encryption algorithms add computation overhead.

Our analysis shows that none of the reviewed frameworks completely align with privacy, scalability, heterogeneity, and autonomy. This is due to inherent trade-offs among them. Mechanisms that strengthen one dimension, e.g. encryption for privacy or centralized control for scalability, often restrict others, such as autonomy or adaptability [Cemri *et al.*, 2025]. Thus, current systems must make explicit design choices based on application requirements. In the future, hybrid agent architectures and adaptive control mechanisms may help balance all four dimensions simultaneously.

6 Conclusions & Future Directions

In this paper, we survey LLM-based agents deployed at the edge and introduce a novel taxonomy along the deployment, functional role, interaction, and adaptation axes. Our observations indicate that LLM-based edge agents have tradeoffs among privacy, scalability, heterogeneity, and autonomy. These results point to interesting research questions:

How can we improve the trade-off between autonomy and security? As autonomy grows at the edge, agents become responsible for critical decisions based on sensitive data. Thus, the necessity for securing the agent becomes critical. This direction is currently not thoroughly explored.

How can we design lightweight LLMs for edge agents that can adapt dynamically in resource-constrained environments? LLM-based edge agents currently cannot maintain strong performance in resource-constrained environments. Architectures that can dynamically adapt to the environment at runtime are critical in enabling efficient LLM-based edge agents.

We need a robust framework that jointly optimizes privacy, scalability, heterogeneity, and autonomy. Addressing this challenge is essential for enabling trustworthy, scalable, and truly autonomous LLM-based edge agents.

References

[Ames *et al.*, 2017] Aaron D. Ames, Xiangru Xu, Jessy W. Grizzle, and Paulo Tabuada. Control barrier function based quadratic programs for safety critical systems. *IEEE Transactions on Automatic Control*, 62(8):3861–3876, 2017.

[Cao *et al.*, 2024] Bowen Cao, Deng Cai, Zhisong Zhang, Yuexian Zou, and Wai Lam. On the worst prompt performance of large language models. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, NIPS '24, Red Hook, NY, USA, 2024. Curran Associates Inc.

[Cemri *et al.*, 2025] Mert Cemri, Melissa Z Pan, Shuyi Yang, Lakshya A Agrawal, Bhavya Chopra, Rishabh Tiwari, Kurt Keutzer, Aditya Parameswaran, Dan Klein, Kannan Ramchandran, Matei Zaharia, Joseph E. Gonzalez, and Ion Stoica. Why do multi-agent LLM systems fail? In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2025.

[Chan *et al.*, 2024] Chi-Min Chan, Weize Chen, Yusheng Su, Jianxuan Yu, Wei Xue, Shanghang Zhang, Jie Fu, and Zhiyuan Liu. Chateval: Towards better LLM-based evaluators through multi-agent debate. In *The Twelfth International Conference on Learning Representations*, 2024.

[Chen *et al.*, 2025] Handi Chen, Weipeng Deng, Shuo Yang, Jinfeng Xu, Zhihan Jiang, Edith CH Ngai, Jiangchuan Liu, and Xue Liu. Towards edge general intelligence via large language models: Opportunities and challenges. *IEEE Network*, 2025.

[Ding *et al.*, 2025] Xianzhong Ding, Zhiyu An, Arya Rathee, and Wan Du. A safe and data-efficient model-based reinforcement learning system for hvac control. *IEEE Internet of Things Journal*, 12(7):8014–8032, 2025.

[Du *et al.*, 2024] Wei Du, Shifei Ding, Lili Guo, Jian Zhang, and Ling Ding. Expressive multi-agent communication via identity-aware learning. In *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence*. AAAI Press, 2024.

[Dwork, 2006] Cynthia Dwork. Differential privacy. In *International colloquium on automata, languages, and programming*, pages 1–12. Springer, 2006.

[European Union, 2016] European Union. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data (General Data Protection Regulation). https://eur-lex.europa.eu/eli/reg/2016/679/oj, 2016. Official Journal of the European Union, L119, 1–88.

[Fu *et al.*, 2024] Wenjie Fu, Huandong Wang, Chen Gao, Guanghua Liu, Yong Li, and Tao Jiang. Membership inference attacks against fine-tuned large language models via self-prompt calibration. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, NIPS '24, Red Hook, NY, USA, 2024. Curran Associates Inc.

[Gill *et al.*, 2024] Sukhpal Singh Gill, Muhammed Golec, Jianmin Hu, Minxian Xu, Junhui Du, Huaming Wu, Guneet Walia, Subramaniam Subramanian Murugesan, Babar Ali, MOHIT KUMAR, Kejiang Ye, Prabal Verma, Surendra Kumar, Félix Cuadrado, and Steve Uhlig. Edge ai: A taxonomy, systematic review and future directions. *Cluster Computing*, 28, 10 2024.

[Hasan *et al.*, 2024] Syed Mhamudul Hasan, Alaa M. Alotaibi, Sajedul Talukder, and Abdur R. Shahid. Distributed threat intelligence at the edge devices: A large language model-driven approach. In *2024 IEEE 48th Annual Computers, Software, and Applications Conference (COMPSAC)*, pages 1496–1497, 2024.

[Hu *et al.*, 2024] Senkang Hu, Zhengru Fang, Zihan Fang, Yiqin Deng, Xianhao Chen, and Yuguang Fang. Agentscodriver: Large language model empowered collaborative driving with lifelong learning. *arXiv preprint arXiv:2404.06345*, 2024.

[Hua *et al.*, 2023] Haochen Hua, Yutong Li, Tonghe Wang, Nanqing Dong, Wei Li, and Junwei Cao. Edge computing with artificial intelligence: A machine learning perspective. *ACM Comput. Surv.*, 55(9), January 2023.

[Jiang *et al.*, 2024] Feibo Jiang, Yubo Peng, Li Dong, Kezhi Wang, Kun Yang, Cunhua Pan, Dusit Niyato, and Octavia A. Dobre. Large language model enhanced multi-agent systems for 6g communications. *Wireless Commun.*, 31(6):48–55, December 2024.

[Karthikeyan *et al.*, 2025] Harish Karthikeyan, Yue Guo, Udari Madhushani Sehwag, Leo de Castro, Antigoni Polychroniadou, Leo Ardon, and Sumitra Ganesh. Agentyxcrypt: Advancing privacy and (secure) computation in AI agent collaboration. In *NeurIPS 2025 Workshop on Regulatable ML*, 2025.

[Kokkonen *et al.*, 2022] Henna Kokkonen, Lauri Lovén, Naser Hossein Motlagh, Abhishek Kumar, Juha Partala, Tri Nguyen, Víctor Casamayor Pujol, Panos Kostakos, Teemu Leppänen, Alfonso González-Gil, et al. Autonomy and intelligence in the computing continuum: Challenges, enablers, and future directions for orchestration. *arXiv preprint arXiv:2205.01423*, 2022.

[Lai *et al.*, 2022] Fan Lai, Yinwei Dai, Sanjay S. Singapuram, Jiachen Liu, Xiangfeng Zhu, Harsha V. Madhyastha, and Mosharaf Chowdhury. FedScale: Benchmarking model and system performance of federated learning at scale. In *International Conference on Machine Learning (ICML)*, 2022.

[Liu *et al.*, 2024] Zechun Liu, Changsheng Zhao, Forrest Iandola, Chen Lai, Yuandong Tian, Igor Fedorov, Yunyang Xiong, Ernie Chang, Yangyang Shi, Raghuraman Krishnamoorthi, Liangzhen Lai, and Vikas Chandra. Mobilellm: optimizing sub-billion parameter language models for on-device use cases. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

[Mao *et al.*, 2024] Jiageng Mao, Junjie Ye, Yuxi Qian, Marco Pavone, and Yue Wang. A language agent for au-

tonomous driving. In *First Conference on Language Modeling*, 2024.

[Martinez-Gil *et al.*, 2025] Jorge Martinez-Gil, Mario Pichler, Nefeli Bountouni, Sotiris Koussouris, Marielena Márquez Barreiro, and Sergio Gusmeroli. *An Agentic Framework for Rapid Deployment of Edge AI Solutions in Industry 5.0*, page 55–68. Springer Nature Switzerland, October 2025.

[McMahan *et al.*, 2017] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

[Mwangi *et al.*, 2025] Agrippina Mwangi, León Navarro-Hilfiker, Lukasz Brewka, Mikkel Gryning, Elena Fumagalli, and Madeleine Gibescu. A threshold-triggered deep q-network-based framework for self-healing in autonomic software-defined iiot-edge networks. *IEEE Transactions on Network and Service Management*, pages 1–1, 2025.

[Ouyang *et al.*, 2025] Tao Ouyang, Guihang Hong, Kongyange Zhao, Zhi Zhou, Weigang Wu, Zhaobiao Lv, and Xu Chen. Adarag: Adaptive optimization for retrieval augmented generation with multilevel retrievers at the edge. In *IEEE INFOCOM 2025 - IEEE Conference on Computer Communications*, pages 1–10, 2025.

[Patil *et al.*, 2024] Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. Gorilla: large language model connected with massive apis. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, NIPS '24, Red Hook, NY, USA, 2024. Curran Associates Inc.

[Salama *et al.*, 2025] Abdelaziz Salama, Mohammed M. H. Qazzaz, Zeinab Nezami, Maryam Hafeez, and Syed Ali Raza Zaidi. Poster: Agentic ai meets neural architecture search: Proactive traffic prediction for ai-ran. In *Proceedings of the 2nd ACM Workshop on Open and AI RAN*, OpenRan '25, page 59–61, New York, NY, USA, 2025. Association for Computing Machinery.

[Semerikov *et al.*, 2025] Serhiy O. Semerikov, Tetiana A. Vakaliuk, Olga B. Kanevska, Oksana A. Ostroushko, and Andrii O. Kolhatin. Edge intelligence unleashed: a survey on deploying large language models in resource-constrained environments. *Journal of Edge Computing*, 4(2):179,Äì233, Nov. 2025.

[Shen *et al.*, 2024] Weizhou Shen, Chenliang Li, Hongzhan Chen, Ming Yan, Xiaojun Quan, Hehong Chen, Ji Zhang, and Fei Huang. Small LLMs are weak tool learners: A multi-LLM agent. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 16658–16680, Miami, Florida, USA, November 2024. Association for Computational Linguistics.

[Shen *et al.*, 2025] Xicong Shen, Yang Liu, Yi Liu, Peiran Wang, Huiqi Liu, Jue Hong, Bing Duan, Zirui Huang, Yunlong Mao, Ye Wu, and Sheng Zhong. Sap: privacy-preserving fine-tuning on language models with split-and-privatize framework. In *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence*, IJCAI '25, 2025.

[Shi *et al.*, 2016] Weisong Shi, Jie Cao, Quan Zhang, Youhuizi Li, and Lanyu Xu. Edge computing: Vision and challenges. *IEEE Internet of Things Journal*, 3(5):637–646, 2016.

[So *et al.*, 2023] Jinhyun So, Ramy E. Ali, Başak Güler, Jiantao Jiao, and A. Salman Avestimehr. Securing secure aggregation: mitigating multi-round privacy leakage in federated learning. In *Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence*. AAAI Press, 2023.

[Wang *et al.*, 2023] Yiming Wang, Yu Lin, Xiaodong Zeng, and Guannan Zhang. Privatelora for efficient privacy preserving llm, 2023.

[Wei *et al.*, 2018] Dennis Wei, Karthikeyan Natesan Ramamurthy, and Kush R. Varshney. Distribution-preserving k-anonymity. *Stat. Anal. Data Min.*, 11(6):253–270, November 2018.

[Woisetschläger *et al.*, 2024] Herbert Woisetschläger, Alexander Erben, Shiqiang Wang, Ruben Mayer, and Hans-Arno Jacobsen. A survey on efficient federated learning methods for foundation model training. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, IJCAI '24, 2024.

[Wu *et al.*, 2024] Shirley Wu, Shiyu Zhao, Qian Huang, Kexin Huang, Michihiro Yasunaga, Kaidi Cao, Vassilis N. Ioannidis, Karthik Subbian, Jure Leskovec, and James Zou. Avatar: optimizing llm agents for tool usage via contrastive reasoning. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, NIPS '24, Red Hook, NY, USA, 2024. Curran Associates Inc.

[Xia *et al.*, 2023] Yuchen Xia, Manthan Shenoy, Nasser Jazdi, and Michael Weyrich. Towards autonomous system: flexible modular production system enhanced with large language model agents. In *2023 IEEE 28th International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–8, 2023.

[Xu *et al.*, 2024] Jiajun Xu, Zhiyuan Li, Wei Chen, Qun Wang, Xin Gao, Qi Cai, and Ziyuan Ling. On-device language models: A comprehensive review. *arXiv preprint arXiv:2409.00088*, 2024.

[Zhang *et al.*, 2021] Meng Zhang, Ermin Wei, and Randall Berry. Faithful edge federated learning: Scalability and privacy. *IEEE Journal on Selected Areas in Communications*, 39(12):3790–3804, 2021.