

Large Language Models over Networks: Collaborative Intelligence under Resource Constraints

Liangqi Yuan, *Graduate Student Member, IEEE*, Wenzhi Fang, *Graduate Student Member, IEEE*, Shiqiang Wang, *Fellow, IEEE*, H. Vincent Poor, *Life Fellow, IEEE*, and Christopher G. Brinton, *Senior Member, IEEE*

Abstract—Large language models (LLMs) are transforming society, powering applications from smartphone assistants to autonomous driving. Yet cloud-based LLM services alone cannot serve a growing class of applications, including those operating under intermittent connectivity, sub-second latency budgets, data-residency constraints, or sustained high-volume inference. On-device deployment is in turn constrained by limited computation and memory. No single endpoint can deliver high-quality service across this spectrum. This article focuses on collaborative intelligence, a paradigm in which multiple independent LLMs distributed across device and cloud endpoints collaborate at the task level through natural language or structured messages. Such collaboration strives for superior response quality under heterogeneous resource constraints spanning computation, memory, communication, and cost across network tiers. We present collaborative inference along two complementary and composable dimensions: vertical device-cloud collaboration and horizontal multi-agent collaboration, which can be combined into hybrid topologies in practice. We then examine learning to collaborate, addressing the training of routing policies and the development of cooperative capabilities among LLMs. Finally, we identify open research challenges including scaling under resource heterogeneity and trustworthy collaborative intelligence.

I. INTRODUCTION

Large language models (LLMs) have demonstrated unprecedented capabilities in question answering, code generation, and multi-modal reasoning, with applications spanning smartphone assistants, autonomous vehicles, and robotic systems. However, these capabilities rest on substantial computational and memory foundations. Frontier LLMs contain hundreds of billions to trillions of parameters, and both training and inference depend on resource-rich cloud data centers. At the same time, a growing number of applications cannot rely on cloud APIs alone. UAVs and field robots may enter connectivity-denied environments, closed-loop control and real-time agents cannot tolerate cloud round-trip latency, regulated domains such as healthcare and finance prohibit sensitive data from leaving the device, and sustained agentic workloads are bounded by per-token pricing and provider rate limits [1]. Lightweight LLMs with hundreds of millions to several billions of parameters make local execution on

L. Yuan, W. Fang, and C. G. Brinton are with Purdue University, IN, USA. E-mail: {liangqi, fang375, cgb}@purdue.edu.

S. Wang is with University of Exeter, UK. E-mail: shiqiang.wang@ieee.org.

H. V. Poor is with Princeton University, USA. E-mail: poor@princeton.edu.

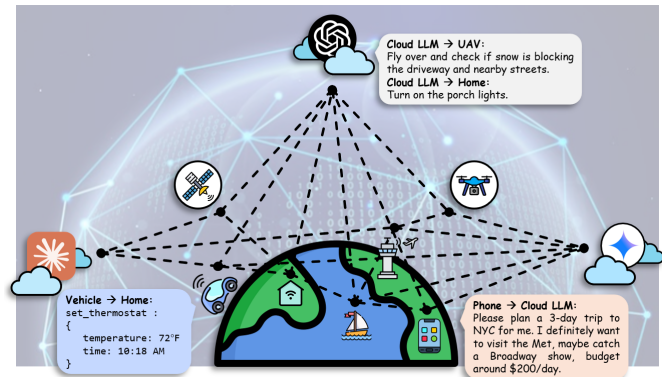


Fig. 1. Collaborative intelligence of LLMs over networks.

smartphones feasible, yet these compact LLMs still exhibit a significant capability gap relative to cloud frontier LLMs on complex tasks.

Prior work on deploying LLMs across network tiers has largely operated at the model level. On the training side, federated learning enables collaborative fine-tuning through gradient or adapter exchange [2]. On the inference side, techniques such as model partitioning, speculative decoding, and context compression reduce the latency or communication cost of serving a single LLM across devices and servers [3]. However, these model-level approaches share common limitations, as they require white-box access to model internals and assume tightly coupled execution across endpoints. More fundamentally, they accelerate a single model rather than orchestrate cooperation among heterogeneous endpoints, including proprietary cloud APIs accessible only through text interfaces, that must jointly handle tasks no single endpoint can complete alone.

This article focuses on a distinct and increasingly important paradigm that we term **collaborative intelligence**, wherein multiple independent LLMs distributed across device and cloud endpoints collaborate at the task level through *semantic exchanges of natural language or structured messages*, rather than model parameters or intermediate tensors. Unlike model-level distribution, each node runs a complete, self-contained LLM instance and exchanges semantic-level information (queries, responses, task descriptions, and observation records) across the network. This design is inherently compatible with black-box API invocation, supports heterogeneous LLM architectures without requiring matched

TABLE I
RESOURCE CONSTRAINTS IN NETWORKED LLM ENVIRONMENTS

Resource	On-Device	Cloud	Impact
Computation	<i>FLOPs, GPU Availability</i> <ul style="list-style-type: none"> • Compute resources shared with other tasks • High per-token latency 	<i>GPU Clusters, Rate Limits</i> <ul style="list-style-type: none"> • Abundant but shared; subject to queuing • Rate limits cap throughput 	<ul style="list-style-type: none"> • Inability to serve time-sensitive applications • Limits concurrent inference requests
Memory	<i>RAM, Storage, KV-Cache</i> <ul style="list-style-type: none"> • Constrained LLM sizes • Restricted KV-cache limits context length 	<i>VRAM, Context Window</i> <ul style="list-style-type: none"> • Memory allocation not user-configurable • Context window capped by provider 	<ul style="list-style-type: none"> • Unable to deploy more capable LLMs • Aggressive quantization degrades output quality
Communication	<i>Bandwidth, Connectivity</i> <ul style="list-style-type: none"> • High round-trip latency • Intermittent and unstable wireless links 	<i>Throughput, Latency</i> <ul style="list-style-type: none"> • Throttled responses under rate limits • Streaming delivery depends on link quality 	<ul style="list-style-type: none"> • Unreliable offloading and orchestration • Restricted real-time inference scenarios
Cost	<i>Energy, Hardware</i> <ul style="list-style-type: none"> • Finite battery limits sustained inference • Capital expense for device compute hardware 	<i>Infrastructure, API Fees</i> <ul style="list-style-type: none"> • Infrastructure fees for cloud compute • Per-token pricing bounds total queries 	<ul style="list-style-type: none"> • Bounded total inference under budget • Sustained high-volume inference economically infeasible

layer configurations, and enables flexible composition of specialized capabilities across nodes. As illustrated in Fig. 1, such collaboration arises naturally in emerging applications, where smartphones offload complex queries to cloud LLMs via natural language, vehicles coordinate with home devices through structured commands, and cloud LLMs orchestrate UAV inspections by exchanging task descriptions with on-device LLMs — all without sharing LLM weights or internal states. Effective collaboration must happen automatically and in real time, jointly navigate quality, latency, communication, and cost under fluctuating conditions, and coordinate context and decisions across endpoints in ways that ad hoc workflows cannot support.

The remainder of this article is organized around five contributions. We introduce collaborative intelligence as a task-level paradigm for networked LLMs, motivated by a resource landscape spanning computation, memory, communication, and cost that makes such collaboration both necessary and difficult (Sec. II). Building on this, we present the system design of collaborative inference along two complementary and composable dimensions, vertical device-cloud and horizontal multi-agent (Sec. III). We then review the learning techniques that equip LLMs to route requests and cooperate effectively (Sec. IV). We further present a case study on device-cloud routing in multi-modal conversations, showing that a learned routing policy with stateful budget tracking breaks the quality-latency-cost tradeoff curve of prompt-based approaches (Sec. V). Finally, we identify open challenges in scaling under heterogeneity and in building trustworthy collaborative systems, and outline directions for future research (Sec. VI).

II. THE LANDSCAPE OF NETWORKED LLMs: CHALLENGES AND OPPORTUNITIES

Today’s LLM ecosystem spans a wide resource spectrum. At one end, lightweight LLMs with a few billion parameters run locally on smartphones and edge devices, offering low latency and no per-query fees, though at the expense of device energy and hardware amortization. At the other

end, cloud-based frontier LLMs with hundreds of billions of parameters deliver state-of-the-art quality but require network connectivity and incur per-token or subscription charges from the service provider [4]. Table I organizes the constraints that govern deployments between these extremes into four categories: computation, memory, communication, and cost. On the device side, limited compute and memory cap model size and context length, while energy and connectivity bound sustained or offloaded inference. On the cloud side, connectivity requirements, round-trip latency floors, provider rate limits, and data-residency rules render cloud-only inference infeasible for an important class of applications, independent of budget. The four categories are also tightly coupled, as quantizing an LLM to fit in device memory sacrifices output quality, while offloading to the cloud trades computation savings for latency and monetary cost. Fig. 2 makes the resulting tradeoff landscape concrete, showing that no single LLM dominates across all dimensions and that substantial heterogeneity exists not only between tiers but also within each tier, where hardware platforms and cloud providers can differ by an order of magnitude in throughput, pricing, and capability.

The binding constraints vary significantly from one device to another, as illustrated in Fig. 3. A smartphone running a quantized LLM is bottlenecked primarily by on-device computation and memory, since it sustains only moderate token throughput and must aggressively quantize to fit within a few gigabytes of RAM, directly limiting local inference quality and context length. A UAV faces a fundamentally different profile, with all resources scarce simultaneously. Limited onboard compute and memory are compounded by intermittent wireless connectivity and a strict energy budget tied to battery capacity and flight time. For such devices, selectively offloading to the cloud can be attractive when connectivity permits, since local inference is itself expensive in both latency and power. Yet the decision must also weigh link reliability and the privacy sensitivity of the transmitted data, which often preclude cloud offloading in mission-critical

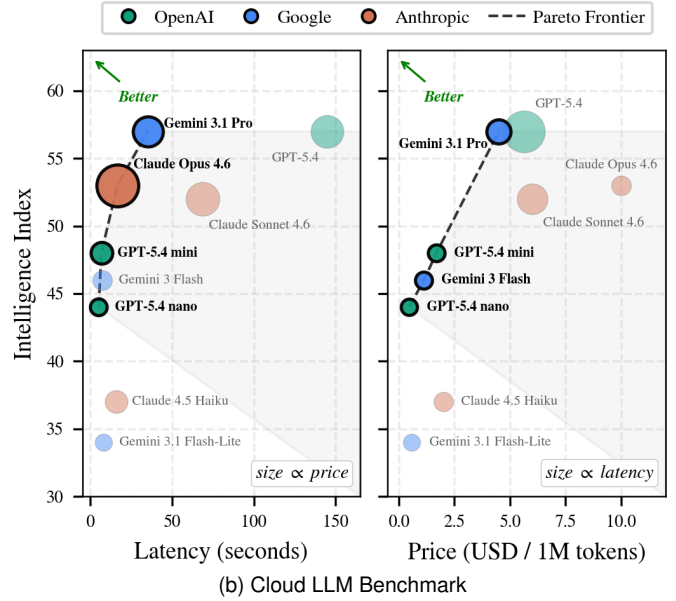
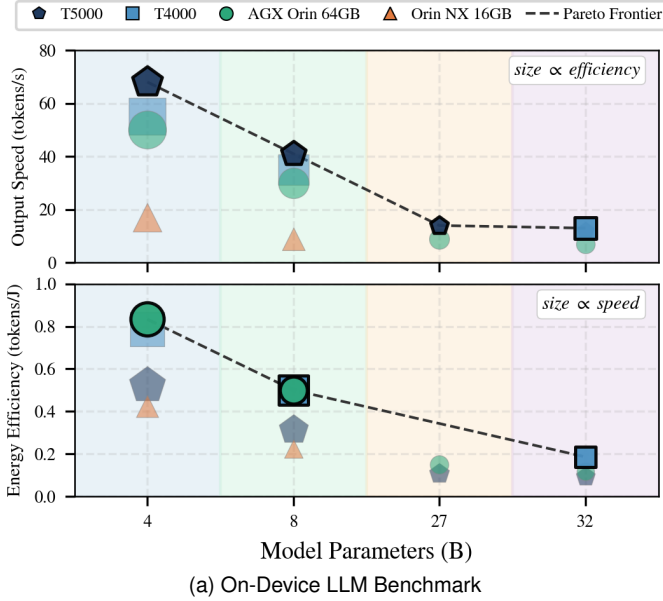


Fig. 2. Performance-resource tradeoffs across network tiers. The Pareto frontier shifts with the binding resource: in (a), the model leading on throughput does not lie on the energy-efficiency Pareto frontier; in (b), cloud models on the latency-quality frontier are no longer Pareto-optimal under price-quality, and vice versa. No single endpoint dominates across all dimensions.²

scenarios. Input modalities differ just as widely, from text and photos on a smartphone to aerial imagery and flight telemetry on a UAV, ruling out any one-size-fits-all strategy.

These observations point to both an opportunity and a set of challenges. Because device and cloud LLMs have complementary strengths, enabling them to collaborate at the task level can deliver a quality-latency-cost balance beyond what any individual endpoint offers. Yet realizing such collaboration is far from straightforward. Routing decisions must be made automatically and continuously, jointly navigating multiple competing objectives rather than optimizing any single one. Multi-turn and agentic workflows compound the difficulty, requiring dialogue history and intermediate observations to be handed off across endpoints. Horizontal cooperation among peer devices adds further design choices around communication topology and message format, which interact with response quality in non-obvious ways. The remaining sections describe how architectural designs and learning techniques address these challenges.

III. COLLABORATIVE INFERENCE ARCHITECTURE

Collaborative inference can be organized along two complementary topologies, illustrated in Fig. 3: vertical device-cloud collaboration, where on-device LLMs offload difficult tasks upward to more capable cloud LLMs, and horizontal multi-agent collaboration, where peer LLM agents collaborate to solve tasks collectively. The two are not mutually exclusive and are often composed into hybrid topologies in practice, as when a cloud LLM coordinates a fleet of on-device agents that also exchange messages directly with one another. Both topologies communicate through natural language or structured messages, with emerging standards such as the

Model Context Protocol (MCP) offering a uniform substrate for such exchanges, while preserving the black-box, task-level interaction that defines collaborative intelligence.

A. Device-Cloud Collaboration

Device-cloud collaboration exists because neither endpoint alone suffices. On-device LLMs lack the capability for complex tasks, while cloud APIs are unavailable or infeasible under resource constraints. The basic premise is therefore to let a lightweight on-device LLM handle requests that it can serve well, and forward the rest to a powerful cloud LLM.

Joint LLM and Modality Selection. For each incoming request, the system must first decide which LLM should serve it. On the demand side, user requests vary in difficulty and domain; a factual lookup is well within reach of a small on-device LLM, while a multi-step reasoning task may require a cloud frontier LLM. On the supply side, available endpoints differ in capability, latency, and cost, and these properties fluctuate with network conditions and server load. When requests additionally involve multi-modal inputs such as images, documents, and sensor data alongside text, a second decision arises, namely which subset of inputs to transmit. Transmitting all available inputs to the cloud is often unnecessary, as the marginal information gain from additional inputs of the same modality diminishes quickly, but which inputs are redundant depends on which LLM will process them. In such cases, jointly optimizing LLM selection and input composition can substantially reduce communication cost while preserving response quality [5].

Context Management across Turns. Many practical applications involve multi-turn conversations or multi-step agentic workflows where context accumulates over time. When the system routes successive turns to different endpoints, the accumulated dialogue history or action trace must be transferred

²Device benchmarks from <https://www.jetson-ai-lab.com/models/>; cloud benchmarks from <https://artificialanalysis.ai/leaderboards/models>.

along with the new query. The overhead is non-trivial, since a ten-turn conversation can easily accumulate thousands of tokens of dialogue history, and agentic workflows that log intermediate observations and tool outputs grow even faster. Retransmitting this context on every endpoint switch consumes both uplink bandwidth and cloud-side prompt tokens the prompt tokens directly increasing monetary cost under per-token pricing. This creates an architectural tension, where frequently switching between on-device and cloud LLMs incurs repeated context transfer overhead, while committing to a single endpoint for an entire session sacrifices the flexibility to adapt as task difficulty evolves across turns. Practical systems address this through context caching and selective summarization, compressing the conversation state before transmission so that cross-endpoint handoffs remain lightweight [6]. Nevertheless, compression is lossy, and how much context to retain versus discard is itself a decision that interacts with routing, since a turn routed to a less capable on-device LLM may need more supporting context than one sent to a frontier cloud LLM.

B. Multi-Agent Collaboration

When a task naturally decomposes into parallel subtasks or benefits from diverse perspectives, multiple LLM agents can collaborate horizontally, each running an independent LLM instance and coordinating through message passing. Such agents may reside on devices, in the cloud, or across both tiers, yielding flexible compositions.

Collaboration Patterns. Multi-agent collaboration can be characterized along several complementary dimensions, such as interaction pattern (e.g., debate, division-of-labor, hierarchical), execution structure (e.g., parallel, sequential), and coordination scope (e.g., centralized, decentralized), each carrying different implications for response quality, latency, communication, and cost [7]. We focus on the first two, which most directly determine how much work the network must carry. Along the interaction axis, three patterns are commonly seen in practice [8]. In debate-style collaboration, agents independently generate responses to the same problem and then broadcast, critique, and revise them over multiple rounds until reaching consensus. In division-of-labor collaboration, a task is decomposed into independent subtasks (e.g., retrieval, computation, and synthesis), each assigned to the agent best equipped for it, with partial results aggregated upon completion. Hierarchical collaboration blends the two, where a supervisor agent decomposes the task and monitors progress while worker agents execute subtasks and report back through structured action-observation logs. Along the execution axis, these patterns unfold differently in time, with division-of-labor exploiting parallelism, hierarchical workflows proceeding sequentially, and debate interleaving both, reflecting that symmetric peer interactions tend toward parallel execution while supervisor-worker structures impose a natural sequential ordering. The right choice depends on task structure and network environment: as illustrated in Fig. 3, a cloud LLM coordinating a UAV fleet maps naturally to hierarchical collaboration, while a group of peer devices with

comparable capabilities, such as smartphones or household robots, is better suited to debate or division-of-labor, where no single node needs a privileged role. These patterns are agnostic to where agents reside: debate may unfold among several cloud LLMs, division-of-labor may split work across a smartphone and a household robot, and hierarchical workflows may mix tiers as in a cloud supervisor directing on-device workers. What defines horizontal collaboration is peer-level message exchange among independent LLM instances, not their physical location.

Communication Topology and Overhead. As the number of collaborating agents grows, the communication topology becomes a critical design choice [9]. A fully connected topology maximizes information sharing but scales quadratically in message volume; a star topology with a central coordinator reduces per-agent communication but creates a bottleneck; a relay or tree topology balances the two. Because each message in collaborative intelligence is a natural language or structured text exchange rather than a compact numerical vector, message length becomes a first-order concern. Debate-style interactions are particularly expensive, since in a fully connected debate of N agents over T rounds, per-round message exchanges grow as $O(N^2)$, and each message itself lengthens across rounds roughly linearly with the number of prior exchanges, yielding total token traffic on the order of $O(N^2T^2)$ in the worst case. By contrast, hierarchical mode is more communication-efficient by design, since only the supervisor exchanges messages with all workers, and structured action-observation logs are typically much shorter than free-form debate responses. Beyond raw overhead, topology choice also shapes the quality of the collective output, as denser connectivity does not always translate into better answers; we return to this interaction in Sec. IV-B. Co-designing the topology and message format to fit the available inter-device bandwidth is essential for making multi-agent collaboration practical on resource-constrained devices.

IV. LEARNING TO COLLABORATE

The preceding section described how LLMs can collaborate; this section examines how those collaboration strategies are learned. The two learning objectives mirror the two collaboration dimensions, where learning to route trains the system to decide which endpoint handles each request in device-cloud settings, while learning to cooperate teaches LLMs to work together effectively in multi-agent settings. The right panel of Fig. 3 summarizes this landscape.

A. Routing Policy Learning

In device-cloud collaboration, every incoming request requires a routing decision between local and cloud processing. Rather than relying on hand-crafted rules, recent work trains routing policies that learn to make this choice by balancing quality, latency, and cost.

Router-Based Selection. A natural starting point is to train a lightweight classifier that examines each incoming query and routes it to the most suitable LLM, balancing quality, latency, and cost [10]. A typical pipeline proceeds

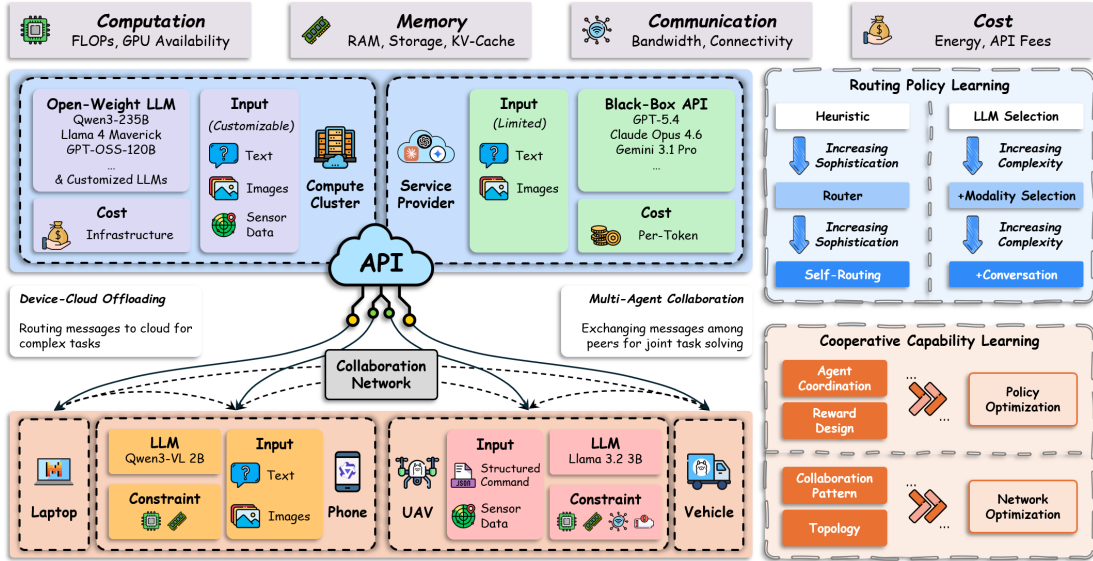


Fig. 3. Collaborative LLM networks: device-cloud offloading and multi-agent collaboration enable heterogeneous LLM endpoints to jointly deliver higher response quality while navigating resource constraints across computation, memory, communication, and cost.

in two stages. Candidate LLMs are first profiled offline to obtain quality scores, average latency, and per-token cost on representative benchmarks. A compact classifier (e.g., a fine-tuned BERT model) then learns to predict, for each incoming query, which candidate offers the best tradeoff. The training objective is usually formulated as a constrained optimization that maximizes expected response quality subject to a budget on average cost or latency. Although such methods generally achieve favorable quality-cost tradeoffs, they treat each query independently and are thus best suited to single-turn settings. In multi-turn conversations, however, routing decisions are coupled across turns, since switching between local and cloud LLMs mid-session requires transferring the accumulated dialogue context, making the routing problem inherently sequential. In this case, the routing policy can be modeled as a Markov decision process in which the state encodes the current query, conversation history, and resource usage so far, and the action selects an endpoint for the current turn. Reinforcement learning then optimizes the cumulative quality-cost tradeoff over an entire conversation rather than a single query [5].

Self-Routing via Post-Training. An alternative eliminates the external router altogether: the on-device LLM is trained to attempt the task first and inspect its own reasoning before deciding whether to escalate [11]. Concretely, the LLM is post-trained using reinforcement learning with a composite reward, where it receives a quality bonus when its local answer is correct and a cost penalty whenever it escalates to the cloud, incentivizing the LLM to handle as many queries locally as possible without sacrificing accuracy. At inference time, the on-device LLM can draw on chain-of-thought signals generated during its own reasoning process, such as self-consistency across sampled traces, entropy of intermediate steps, or explicit self-evaluation tokens, to gauge whether the current problem lies within its capability. Because the routing decision is made after the LLM has already engaged with the

problem rather than from surface-level features of the prompt alone, it yields substantially more accurate offloading than classifier-based routers.

B. Cooperative Capability Learning

Most existing multi-agent LLM systems rely on prompt engineering to drive collaboration, assigning roles and instructions without any cooperative training. Yet LLMs are pre-trained in isolation and have never experienced genuine multi-party interaction, limiting what static prompts alone can achieve. Closing this gap requires fine-tuning agents to collaborate explicitly.

Cooperative Policy Optimization. The most direct approach is to move beyond static prompts and explicitly train LLMs for collaboration. Current prompt-based systems assign roles and instructions but leave agents to improvise their interaction strategies, often resulting in repetitive exchanges that fail to converge or even degrade performance over additional rounds. Training-based methods address this by placing multiple LLMs in authentic collaborative scenarios and optimizing their behavior through reinforcement learning. A key finding in this line of work is that training agents in isolation is insufficient, because an agent optimized against a fixed partner tends to act independently rather than cooperate, since it never learns to adapt to evolving strategies [12]. Genuine collaborative behavior emerges only when all participating agents are co-trained simultaneously, allowing each to adjust its policy in response to others. Moreover, cloud LLMs can play an active role in multi-agent collaboration beyond simply answering offloaded queries, where a capable cloud LLM can coordinate device-side agents by deciding which agents to activate and when to conclude the discussion, learning such coordination policies through reinforcement learning [13].

Inter-Agent Network Optimization. While cooperative training determines the content of communication among agents, an equally important question is how to enable such

communication efficiently under bandwidth and computational constraints. One line of work tackles this at the topology level by replacing the default fully connected graph with a sparse communication topology so that each agent only observes a subset of peers’ responses in each round. Empirical results show that such sparse configurations can significantly reduce token costs while matching or even exceeding the performance of fully connected debate, because limiting each agent’s view filters out redundant information and encourages more independent reasoning [14]. A complementary line of work operates at the message level by incorporating communication cost directly into the training objective, penalizing excessive rounds and low-information messages, so that agents learn to convey more concise and informative content per exchange, reaching effective consensus in fewer turns [15].

V. CASE STUDY: DEVICE-CLOUD ROUTING IN MULTI-MODAL CONVERSATIONS

Consider a kitchen assistant scenario, where a user interacts with a smartphone-class device in a kitchen environment and issues a stream of requests spanning four task categories of varying difficulty, from simple message editing to scene-grounded queries that require visual context [5]. Three image modalities are available from wearable and environmental cameras (first-person, overhead, and side views), each capturing complementary but partially redundant information. The on-device endpoint runs a lightweight text-only LLM (Phi-3-mini) on Jetson TX2-class hardware, while the cloud endpoint exposes a multi-modal frontier LLM (GPT-4o) through a paid API. The user specifies a budget of 30 seconds of cumulative latency and \$0.05 USD of cumulative cost for the entire conversation, within which the system must maximize response quality. At each turn, a learned routing policy selects the endpoint and, when the cloud is chosen, a subset of the three modalities to transmit. A key refinement over the single-turn router described earlier is that the remaining budget is included as part of the policy’s state, allowing it to reason about when to spend resources over the course of a conversation rather than whether to spend them on average.

Fig. 4 compares this learned router against representative baselines. On-device-only inference is fast and free but yields the lowest response quality, while cloud-only inference achieves the highest quality at the cost of frequently violating the budget. The LLM-as-Router baselines, which prompt off-the-shelf LLMs to make routing decisions, trace out a clear quality-latency tradeoff, where higher quality comes only at the price of proportionally higher latency and cost. The RL router breaks out of this tradeoff curve, achieving comparable response quality at substantially lower latency and cost. This gap is structural, since prompt-based LLM routers lack a mechanism to track cumulative resource consumption across turns, whereas the learned policy internalizes the remaining budget as part of its state and distributes spending accordingly. These observations suggest that the central value of learned routing lies less in per-query classification than in reasoning about resource budgets as a stateful, temporal constraint.

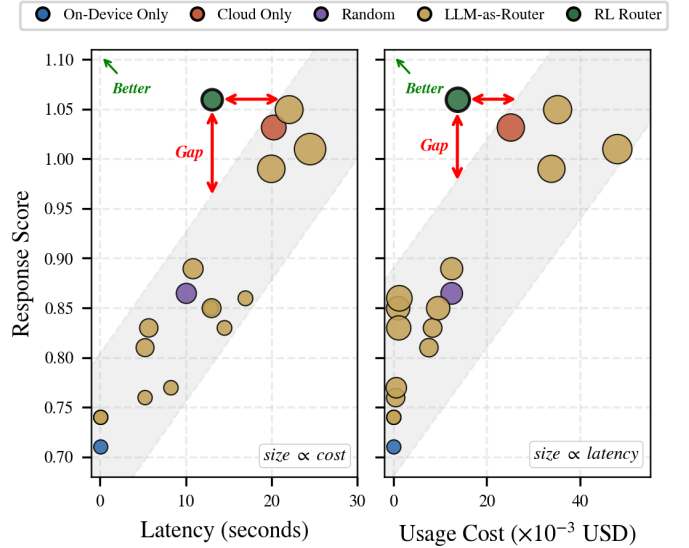


Fig. 4. Quality-latency-cost tradeoffs across device-cloud routing strategies in a multi-modal conversational setting. Baseline methods trace a common tradeoff band (shaded) where higher response quality comes only at proportionally higher latency and cost. The RL-based learned router escapes this band in both views by internalizing cumulative resource consumption as part of its state rather than deciding per query.

VI. OPEN CHALLENGES AND RESEARCH DIRECTIONS

Deploying collaborative LLM systems over networks at scale exposes a range of unsolved problems. Below we highlight two broad axes of open challenges and identify promising directions for each.

A. Scaling Under Resource Heterogeneity

Adapting to constantly shifting ecosystems: The resource spectrum that makes collaboration attractive also makes it difficult to engineer. A single collaboration framework must accommodate endpoints whose compute, memory, bandwidth, and cost budgets differ by orders of magnitude, as outlined in Table I, and these gaps will only widen as frontier cloud LLMs continue to grow while on-device deployment pushes toward ever smaller form factors. Today’s solutions typically target a fixed set of LLMs and hardware profiles; an important open question is how to build collaboration protocols that remain effective as new LLMs and devices continuously enter and leave the ecosystem, without requiring retraining or manual reconfiguration each time.

From pairwise collaboration to large-scale swarms: Most existing work studies collaboration among a handful of agents. Scaling to hundreds or thousands of LLM instances, such as a cloud coordinator managing a fleet of on-device agents in autonomous vehicles or drone swarms, raises challenges at every level. Scheduling must handle continuous streams of inference requests under strict latency budgets, agents must reach consensus over volatile and bandwidth-limited links, and individual nodes must reason autonomously during connectivity gaps. At the same time, many high-value tasks require maintaining coherent context across dozens of inference steps, and serving such long-context workloads concurrently for many agents imposes severe memory and communication

pressure. How to co-design scheduling, communication, and fault-tolerance mechanisms that operate reliably across all these levels remains largely unexplored.

B. Trustworthy Collaborative Intelligence

Keeping sensitive data under control: Collaborative inference inherently moves queries and intermediate results across trust boundaries. User prompts may carry personally identifiable information, medical records, or financial data, and agentic LLMs with access to local files and system interfaces can inadvertently leak private content into their responses and propagate it across a multi-agent chain. The tension is between the openness needed for effective collaboration and the containment needed for privacy, since sharing more context generally improves response quality, but every additional piece of information transmitted is a potential exposure. Balancing this tradeoff, rather than defaulting to one extreme, is an important direction for future work.

Defending against cascading attacks: Prompt injection is well studied as a single-interface threat, but distributed LLM deployments introduce a new dimension, where a successful injection at any node can propagate along the collaboration hierarchy and contaminate downstream agents across the entire workflow. Injected instructions can persist covertly within accumulated context, interfering with subsequent inference steps long after the initial attack. The risk is amplified when LLM agents directly control physical systems such as vehicles or robots, where a compromised output can cause real-world harm. Understanding how adversarial inputs propagate through multi-tier collaborative systems, and developing defenses that can detect and contain such cascading effects in real time, is a critical open problem.

VII. CONCLUSION

As LLMs become embedded in every tier of the network, no single endpoint can meet the full spectrum of user demands on its own. Collaborative intelligence enables heterogeneous LLMs to cooperate at the task level through natural language and structured messages, achieving quality-latency-cost tradeoffs beyond any individual model. This article has organized the landscape along two complementary dimensions, device-cloud offloading and multi-agent collaboration, and examined how routing policies and cooperative training can make such collaboration effective. Realizing this vision at scale will require progress on several fronts, from protocols that adapt seamlessly as models and devices enter and leave the ecosystem, to trust and safety mechanisms that keep pace with the growing autonomy of LLM agents. We believe collaborative intelligence will become a foundational design principle for networked AI systems in the years ahead.

ACKNOWLEDGMENT

The authors thank Dr. Dusit Niyato for his insightful feedback on this paper.

REFERENCES

- [1] Y. Zheng, Y. Chen, B. Qian, X. Shi, Y. Shu, and J. Chen, "A Review on Edge Large Language Models: Design, Execution, and Applications," *ACM Computing Surveys*, vol. 57, no. 8, pp. 1–35, 2025.
- [2] Y. Cheng, W. Zhang, Z. Zhang, C. Zhang, S. Wang, and S. Mao, "Toward Federated Large Language Models: Motivations, Methods, and Future Directions," *IEEE Communications Surveys & Tutorials*, vol. 27, no. 4, pp. 2733–2764, 2024.
- [3] Z. Lin, G. Qu, Q. Chen, X. Chen, Z. Chen, and K. Huang, "Pushing Large Language Models to the 6G Edge: Vision, Challenges, and Opportunities," *IEEE Communications Magazine*, vol. 63, no. 9, pp. 52–59, 2025.
- [4] G. Qu, Q. Chen, W. Wei, Z. Lin, X. Chen, and K. Huang, "Mobile Edge Intelligence for Large Language Models: A Contemporary Survey," *IEEE Communications Surveys & Tutorials*, vol. 27, no. 6, pp. 3820–3860, 2025.
- [5] L. Yuan, D.-J. Han, S. Wang, and C. Brinton, "Local-Cloud Inference Offloading for LLMs in Multi-Modal, Multi-Task, Multi-Dialogue Settings," in *Proceedings of the Twenty-Sixth International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing*, 2025, pp. 201–210.
- [6] H. Jiang, Q. Wu, C.-Y. Lin, Y. Yang, and L. Qiu, "LLMLingua: Compressing Prompts for Accelerated Inference of Large Language Models," in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023, pp. 13 358–13 376.
- [7] H. Luo, Y. Liu, R. Zhang, J. Wang, G. Sun, D. Niyato, H. Yu, Z. Xiong, X. Wang, and X. Shen, "Toward Edge General Intelligence with Multiple-Large Language Model (Multi-LLM): Architecture, Trust, and Orchestration," *IEEE Transactions on Cognitive Communications and Networking*, vol. 11, no. 6, pp. 3563–3585, 2025.
- [8] Q. Wu, G. Bansal, J. Zhang, Y. Wu, B. Li, E. Zhu, L. Jiang, X. Zhang, S. Zhang, J. Liu *et al.*, "AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation," in *Proceedings of the First Conference on Language Modeling*, 2024.
- [9] G. Zhang, Y. Yue, Z. Li, S. Yun, G. Wan, K. Wang, D. Cheng, J. X. Yu, and T. Chen, "Cut the Crap: An Economical Communication Pipeline for LLM-based Multi-Agent Systems," in *Proceedings of the Thirteenth International Conference on Learning Representations*, 2025.
- [10] D. Ding, A. Mallick, C. Wang, R. Sim, S. Mukherjee, V. Rühle, L. V. Lakshmanan, and A. H. Awadallah, "Hybrid LLM: Cost-Efficient and Quality-Aware Query Routing," in *Proceedings of the Twelfth International Conference on Learning Representations*, 2024.
- [11] W. Fang, D.-J. Han, L. Yuan, E. Chen, and C. Brinton, "Bridging On-Device and Cloud LLMs for Collaborative Reasoning: A Unified Methodology for Local Routing and Post-Training," in *Proceedings of the Forty-third International Conference on Machine Learning*, 2026.
- [12] C. Park, S. Han, X. Guo, A. E. Ozdaglar, K. Zhang, and J.-K. Kim, "MAPoRL: Multi-Agent Post-Co-Training for Collaborative Large Language Models with Reinforcement Learning," in *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2025, pp. 30 215–30 248.
- [13] Y. Dang, C. Qian, X. Luo, J. Fan, Z. Xie, R. Shi, W. Chen, C. Yang, X. Che, Y. Tian *et al.*, "Multi-Agent Collaboration via Evolving Orchestration," in *Proceedings of the Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.
- [14] Y. Li, Y. Du, J. Zhang, L. Hou, P. Grabowski, Y. Li, and E. Ie, "Improving Multi-Agent Debate with Sparse Communication Topology," in *Findings of the Association for Computational Linguistics: EMNLP 2024*, 2024, pp. 7281–7294.
- [15] W. Chen, J. Yuan, C. Qian, C. Yang, Z. Liu, and M. Sun, "Optima: Optimizing Effectiveness and Efficiency for LLM-Based Multi-Agent System," in *Findings of the Association for Computational Linguistics: ACL 2025*, 2025, pp. 11 534–11 557.

Liangqi Yuan is pursuing the Ph.D. degree in ECE with Purdue University.

Wenzhi Fang is pursuing the Ph.D. degree in ECE with Purdue University.

Shiqiang Wang is a Professor of Artificial Intelligence in the Department of Computer Science, University of Exeter.

H. Vincent Poor is the Michael Henry Strater University Professor in ECE with Princeton University.

Christopher G. Brinton is the Elmore Associate Professor of ECE with Purdue University.