Journal of Artificial Intelligence for Automation
https://www.sciltp.com/journals/jaia

Check for updates

Article

Towards a Unified Framework for Large–Small Model Collaboration in Cloud–Edge Systems

Jing Bi [1], Yaxi Yang [1], Haitao Yuan [2,*], Ziqi Wang [1], Jiayi Li [1], Jiahui Zhai [1] and Jia Zhang [3]

[1] College of Computer Science, Beijing University of Technology, Beijing 100124, China
[2] School of Automation Science and Electrical Engineering, Beihang University, Beijing 100191, China
[3] Department of Computer Science, Lyle School of Engineering, Southern Methodist University, Dallas, TX 75205, USA
* Correspondence: yuan@buaa.edu.cn

How To Cite: Bi, J.; Yang, Y.; Yuan, H.; et al. Towards a Unified Framework for Large–Small Model Collaboration in Cloud–Edge Systems. *Journal of Artificial Intelligence for Automation* **2026**, *1*(3), 12. https://doi.org/10.53941/jaia.2026.100012

Received: 9 March 2026
Revised: 20 June 2026
Accepted: 10 July 2026
Published: 17 July 2026

Abstract: Foundation models have improved the reasoning and generation ability of artificial intelligence systems. However, they are difficult to deploy in edge environments with limited computation, memory, and data access. Small models are easier to run on edge devices. They support fast and low-latency inference, but they often lack global semantic reasoning and cross-domain generalization. This gap between model ability and deployment cost motivates large–small model collaboration in cloud–edge systems. This survey provides a systematic review and a knowledge-flow-oriented taxonomy of such collaboration. It focuses on how cloud-side large models and edge-side small models share, update, and coordinate knowledge. We review knowledge distillation, split inference, federated and continual adaptation, and elastic offloading. We also cover lightweight deployment, modular expert design, privacy-aware coordination, and agent-driven orchestration. Unlike surveys on edge intelligence, federated learning, model compression, TinyML, or cloud–edge resource scheduling, this survey centers on model collaboration. We treat large–small collaboration as a knowledge-centered problem linked to real deployment constraints. We further discuss trade-offs in accuracy, latency, bandwidth, privacy, energy efficiency, adaptability, and lifecycle management. Finally, we identify open challenges for trustworthy, sustainable, and self-evolving cloud–edge collaborative intelligence.

Keywords: large–small model collaboration; artificial intelligence; cloud–edge collaboration; agent-driven orchestration; and deep learning

1. Introduction

Large foundation models have changed how artificial intelligence (AI) systems are built. Many AI systems are no longer limited to narrow, task-specific pipelines. They can now support reasoning, generation, and multimodal understanding across different domains [1]. However, these models are hard to deploy at the edge. They require high computation, large memory, and frequent data access. This creates a clear gap between model capability and deployment feasibility.

Cloud–edge intelligence offers a practical setting for this gap. Edge devices are close to data sources. They can support fast sensing, local inference, and timely response. Yet small edge models usually lack global semantic reasoning and cross-domain generalization. Cloud-side large models provide broader knowledge and stronger reasoning. Edge-side small models provide low latency and local adaptation. Their complementary roles make large–small model collaboration important for cloud–edge systems [2].

A fully centralized AI architecture is difficult to sustain in many real systems. Large models increase inference delay, energy use, and carbon cost [3]. They also depend on data transfer to the cloud. This creates privacy, security, and compliance concerns. These concerns are common in healthcare, autonomous driving, and industrial Internet of

Things (IoT). Robustness is another issue. Edge environments often change quickly. Devices differ in hardware, networks, and local data. A static cloud model may not adapt fast enough. These issues show that cloud intelligence and edge execution must be coordinated more closely [4].

Several mechanisms have been studied for this purpose. Knowledge distillation can transfer knowledge from a large teacher to a small student. Split inference can divide computation between the cloud and edge. Federated learning can train models without moving raw data. Continual adaptation can help models respond to data drift. These methods support knowledge flow between large and small models [5]. Still, each method solves only part of the problem. A complete view should consider accuracy, latency, bandwidth, privacy, energy, and model updates together.

Several recent surveys are related to this topic. They cover edge intelligence, federated learning, TinyML, model compression, cloud–edge computing, and distributed AI [6, 7]. These surveys provide useful views on resource management, communication cost, privacy, and lightweight deployment. However, their main focus is often system-level design. They usually ask where computation should run, how communication can be reduced, or how distributed models can be aggregated. In this survey, we focus on a different question. We study how large and small models exchange knowledge, keep semantic consistency, adapt over time, and work under real deployment constraints. This view treats large–small model collaboration as both a knowledge-flow problem and a deployment problem.

Table 1 shows the position of this survey. It compares related surveys in terms of research focus, large–small model collaboration, knowledge flow, agent-based coordination, and deployment guidance.

Table 1. Positioning of this survey relative to representative surveys on edge intelligence, federated learning, model compression, TinyML, and cloud–edge computing. Bold text highlights the row corresponding to this survey.

Survey	Main Focus	Large–Small Model Collaboration	Knowledge Flow and Lifecycle	Agent-Driven Orchestration	Deployment-Oriented Guidance
Wang et al. [8]	End–edge–cloud collaborative computing for deep learning	Related through collaborative training, inference, and updating	Discussed from the broader end–edge–cloud view	Not a main coordination mechanism	Gives general design guidance for end–edge–cloud deep learning systems
Hoffpauir et al. [9]	Edge intelligence applications and lightweight machine learning	Focuses mainly on lightweight edge models	Cross-tier knowledge evolution is only partly covered	Outside the main scope	Gives guidance for edge-side lightweight AI deployment
Kairouz et al. [10]	Federated learning, privacy-preserving training, and open problems	Indirectly related through distributed training	Focuses on training, aggregation, and privacy	Not a central design issue	Discusses privacy, communication, and FL system constraints
Cheng et al. [11]	DNN model compression and acceleration	Related through compression and teacher–student distillation	Mainly covers compression-based knowledge transfer	Outside the survey scope	Gives guidance for compact DNN deployment
Alajlan and Ibrahim [12]	TinyML inference on low-power IoT devices	Focuses on small models and embedded inference	Cloud–edge knowledge exchange is not a main focus	Outside the survey scope	Gives hardware-aware guidance for ultra-low-power edge deployment
This survey	**Large–small model collaboration in cloud–edge systems**	**Central research focus**	**Analyzed through knowledge transfer, adaptation, synchronization, and lifecycle management**	**Discussed as an emerging coordination layer**	**Analyzes strategy choice under latency, bandwidth, privacy, energy, and device constraints**

Among these studies, Wang et al. [8] is the closest to our topic. It reviews end–edge–cloud collaborative computing for deep learning. Our survey has a more specific focus. We study large–small model collaboration as a model-level and knowledge-flow problem. We emphasize knowledge exchange, semantic consistency, lifecycle adaptation, and agent-based coordination under cloud–edge deployment constraints.

Based on this focus, this paper provides a systematic survey, a knowledge-flow-oriented taxonomy, and a design-oriented analysis of large–small model collaboration in cloud–edge intelligent systems. The main contributions are as follows:

- We review key mechanisms for large–small model collaboration. These include knowledge distillation, split inference, federated and continual adaptation, lightweight deployment, modular expert design, and privacy-aware coordination.
- We organize these mechanisms from a knowledge-flow view. This links model-level collaboration with orchestration, resource scheduling, lifecycle management, and trustworthy deployment.
- We discuss agent-driven orchestration as an emerging coordination layer. It can support adaptive planning, semantic alignment, and autonomous cloud–edge coordination.

- We analyze major collaboration strategies under practical constraints. These include latency, bandwidth, privacy, energy use, device capability, adaptability, and synchronization cost.
- We summarize open challenges for trustworthy, sustainable, and self-evolving cloud–edge collaborative intelligence.

Figure 1 gives an overview of the cloud–edge large–small collaboration paradigm. Section 2 defines the scope of large–small model collaboration. It also reviews the evolution of cloud–edge–end intelligence. Section 3 discusses the main collaboration mechanisms. Section 4 presents the cloud–edge collaborative intelligence framework. Section 5 discusses agent-enhanced collaboration and autonomy. Section 6 summarizes open challenges and future directions. Section 7 concludes the paper.

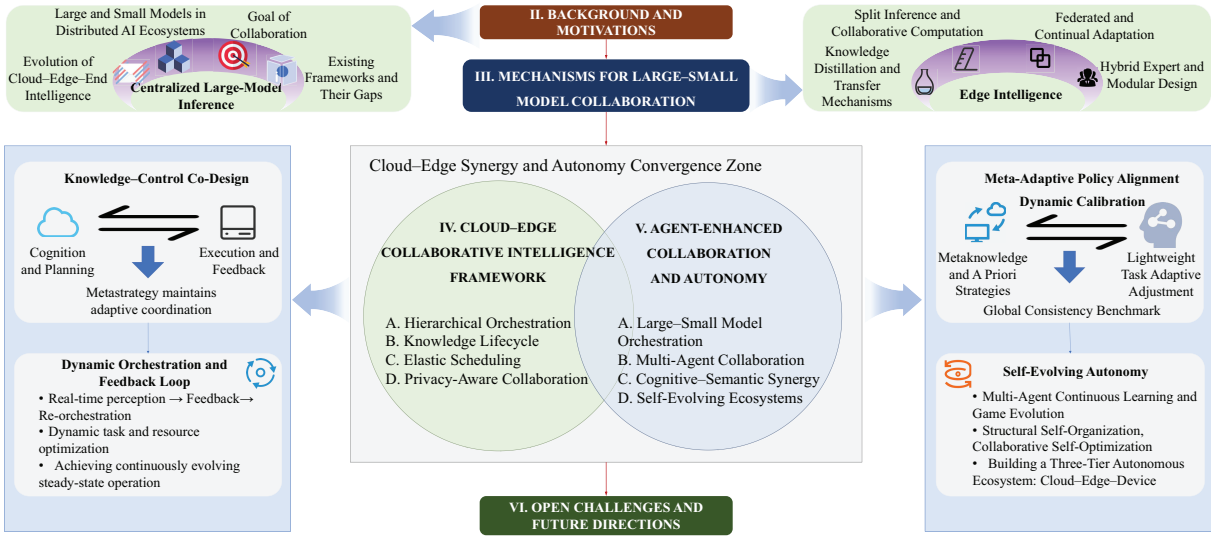

Figure 1. Conceptual evolution from centralized AI to cloud–edge–end collaborative intelligence. The figure shows how large and small models interact through knowledge flow, feedback, and agent-driven orchestration. The Knowledge–Control Co-Design loop highlights the link between cloud-side cognition and edge-side execution feedback.

2. Background and Motivations

2.1. Definitions and Scope of Large–Small Model Collaboration

This section first clarifies three basic terms: “large models”, “small models”, and “collaboration”. These terms are used in the context of cloud–edge intelligent systems.

In this survey, a large model refers to a high-capacity foundation model or task-general model. It is usually deployed in the cloud or on powerful edge servers. Such models often contain billions, tens of billions, or even hundreds of billions of parameters. They provide global semantic reasoning, cross-domain generalization, multimodal understanding, and long-term knowledge abstraction. Typical examples include large language models, vision–language models, large vision transformers, and multimodal foundation models [13,14]. Thus, large models mainly serve as sources of high-level reasoning and general knowledge.

A small model refers to a compact, task-specific, or compressed model. It can run on edge or end devices under limits of latency, memory, energy, and bandwidth. Small models include lightweight convolutional networks, compact vision transformers, and compressed language or multimodal models. Some may contain fewer than one billion parameters. Others may have a few billion parameters but remain deployable after compression. In practice, model size is not defined by parameter count alone. FLOPs, activation memory, quantization precision, runtime framework, latency budget, and power use also matter. Hardware is another key factor. Edge GPUs, NPUs, industrial gateways, mobile AI accelerators, and embedded platforms all impose different limits. For example, Jetson-class GPUs and specialized NPUs can support larger edge models than microcontrollers. Their memory, precision support, software stack, and power budget jointly determine whether a model is practically “small” [15,16]. In this sense, small models emphasize deployability, fast execution, and local adaptation.

Large–small model collaboration refers to more than compressing a large model into a small one. In this survey, it means model-level and system-level coordination between cloud-side large models and edge-side small models. This coordination may occur during training, inference, adaptation, deployment, and lifecycle management. It includes knowledge transfer, split inference, federated and continual adaptation, quantization, pruning, modular expert collaboration, resource scheduling, elastic offloading, and agent-driven orchestration [17]. The key question is how to build a coordinated cloud–edge pipeline. In this pipeline, large models provide semantic priors and global

reasoning. Small models provide efficient local execution and contextual feedback. The two sides should adapt together under real deployment constraints.

Following recent discussions on on-device AI models and edge large language models [15, 16], Table 2 summarizes an indicative hardware-aware definition of large and small models. These thresholds should not be viewed as fixed boundaries. Instead, they reflect practical deployment conditions in cloud–edge systems, where model size must be interpreted together with FLOPs, memory footprint, quantization precision, latency budget, power consumption, and hardware acceleration capability.

Table 2. Hardware-aware definition of large and small models in cloud–edge collaboration. The thresholds are indicative rather than absolute, since practical deployability depends on parameter count, FLOPs, memory footprint, quantization precision, latency budget, and hardware support.

Model Category	Indicative Scale	Typical Deployment Tier	Representative Hardware	Main Deployment Constraints
Cloud-side large foundation model	Tens to hundreds of billions of parameters, e.g., >70 B parameters	Cloud data center or powerful edge cloud	GPU clusters, cloud accelerators, or high-memory AI servers	High memory demand, high serving cost, high energy consumption, and data transfer overhead
Edge-server large or medium model	Several billion to tens of billions of parameters, often quantized or sparsely activated	Edge server, industrial gateway, or regional edge cloud	Edge GPUs, server-grade NPUs, or AI acceleration cards	Limited memory, latency constraints, model loading cost, and scheduling overhead
Lightweight small model	Usually below a few billion parameters, e.g., <3 B parameters, or larger models compressed by pruning, distillation, or INT8/INT4 quantization	Edge device, mobile device, robot, vehicle, or smart camera	Jetson-class GPUs, mobile NPUs, embedded AI chips, or industrial controllers	Real-time latency, power budget, activation memory, thermal limits, and local adaptation capability
Tiny or embedded model	Millions to hundreds of millions of parameters, depending on the task and hardware	End device, sensor node, microcontroller, or low-power IoT device	Microcontrollers, TinyML platforms, low-power DSPs, or lightweight NPUs	Extremely limited memory, intermittent connectivity, battery lifetime, and simplified model update mechanisms

2.2. Goal of Collaboration

Traditional AI systems often rely on centralized cloud inference. Computation, model updates, and decisions are handled in large data centers. This design benefits from strong computation and unified data access. However, it also brings high latency, heavy bandwidth use, and weak response to local changes. As AI moves into cyber–physical systems, real-time and context-aware decisions become more important. This need drives the shift from cloud-centered intelligence to cloud–edge collaboration [18].

Large and small models play different roles in this shift. Large foundation models, such as large language models and vision transformers, provide global reasoning and cross-domain generalization. Small edge models provide efficient and low-latency inference under resource limits [19]. Yet neither side is sufficient alone. Large models are often too expensive for direct edge deployment. Small models are efficient, but they lack deep semantic reasoning and cross-scenario adaptation. This capability–resource gap creates the need for large–small model collaboration [20]. In this setting, the cloud maintains global knowledge, while the edge handles local perception and fast response.

Real systems further strengthen this need. Relying only on cloud-hosted large models can cause high energy use and unacceptable delay. This is risky for time-sensitive tasks such as industrial control and autonomous driving. At the same time, privacy rules and data-locality requirements limit raw data transfer to the cloud. Purely edge-based inference also has limits. It may fail to keep global consistency or handle unseen contexts. Therefore, scalable cloud–edge intelligence must rely on coordination and knowledge exchange between large and small models. This makes collaboration a core structure for future distributed AI systems [21].

2.3. Existing Frameworks and Their Gaps

Several existing paradigms are closely related to this topic. Edge AI, Federated AI, and Tiny Machine Learning (TinyML) have improved latency, energy efficiency, and privacy protection. Still, they mainly focus on systems or resources. They do not fully support model-level synergy between cloud-side large models and edge-side small models. In particular, they offer limited support for knowledge continuity, semantic alignment, and lifecycle co-evolution across tiers.

Edge AI mainly studies computation offloading and resource-aware inference. Fine-grained Deep Neural Network (DNN) partitioning [22] and dynamic model scaling [23] can reduce latency under changing workloads. Other offloading frameworks [24–26] improve execution efficiency and system welfare. However, these methods often treat models as computational workloads. They focus on where inference should run. They pay less attention

to how knowledge is shared or updated across cloud and edge. As a result, cross-tier learning remains fragmented and sensitive to model drift.

Federated AI focuses on privacy-preserving distributed training. Recent federated fine-tuning methods for foundation models [27] reduce communication through parameter-efficient adaptation. Low-Rank Adaptation (LoRA) is one common example. However, many federated methods align parameters rather than semantic representations. Non-IID edge data can also cause semantic divergence among local models. Earlier resource-aware methods [24,25] improve fairness and efficiency. But they do not fully address cross-tier expert consistency. Thus, federated learning still struggles to maintain long-term semantic coherence between large and small models.

TinyML represents the most resource-constrained setting. It enables ultra-low-power inference on microcontrollers. Some works combine Federated Learning (FL) with TinyML [28]. They use OTA updates and compression to improve communication efficiency. However, end devices still have limited representation capacity and unstable connectivity. These limits weaken bidirectional knowledge exchange. In many cases, TinyML devices remain local sensing endpoints rather than active participants in collaborative learning.

These paradigms each solve an important part of the problem. Edge AI decides where computation should run. Federated AI studies how distributed models can be trained. TinyML reduces resource use on small devices. However, none of them fully explains how heterogeneous models can co-evolve. Robust large–small model collaboration needs a broader structure. A knowledge plane is needed for semantic alignment and distillation. A control plane is needed for orchestration and synchronization. A data plane is needed for telemetry, drift monitoring, and version management. With these planes, cloud and edge models can update more coherently over time.

3. Mechanisms for Large–Small Model Collaboration

3.1. Knowledge Distillation and Transfer Mechanisms

Knowledge distillation (KD) is a common way to reduce the gap between large and small models. A cloud-side teacher can provide richer knowledge. An edge-side student can then learn compact representations for local inference. This helps preserve task accuracy while reducing computation and communication cost. For this reason, KD is a basic mechanism for large–small model collaboration.

In this survey, KD [29] refers to teacher–student knowledge transfer. Federated distillation (FD) [30] is treated as its distributed and privacy-aware extension. KD transfers logits, features, attention maps, or intermediate representations. The transfer is usually from a cloud-side teacher to a compact edge-side student. FD extends this idea to multiple edge clients. It exchanges soft labels, prototypes, logits, or representations instead of raw data or full gradients. Thus, KD mainly reduces the capacity gap between large and small models. FD also addresses client heterogeneity, non-IID data, privacy limits, and communication-efficient aggregation.

Early cloud–edge KD studies mainly focus on task compression. KeepEdge [31] uses cloud teachers to guide edge students for real-time UAV inference. It targets strict latency and energy limits. Chen et al. [32] use selective feature distillation. Their method maps multi-branch cloud models into efficient edge encoders. Zhang et al. [33] show that attention-guided and non-local feature alignment can improve lightweight object detectors. These studies show the value of feature transfer and intermediate-layer alignment.

KD has also moved beyond one-time compression. ASMAFL [34] adds periodic re-distillation to asynchronous federated updates. This helps reduce model drift. Adaptive partitioning methods [35] adjust teacher–student roles according to networks and workloads. These methods support cross-tier consistency in changing environments.

Recent work further extends KD to multimodal and modality-asymmetric settings. Selective multimodal-to-unimodal distillation [32] keeps semantic consistency when some edge sensors are missing. MASA [36] uses cross-modal local distillation to align heterogeneous clients. These studies show a shift from simple compression to semantic alignment. Table 3 summarizes representative KD studies. They can be grouped into task-level distillation, hierarchical or continual distillation, and cross-modal distillation.

Discussion and design implications. KD is most useful when the cloud teacher is reliable. It also needs representative calibration data. KD can reduce edge computation and communication cost. But its performance depends on teacher quality and teacher–student alignment [37]. Under strong distribution shift [38], the student may learn biased or incomplete knowledge. Therefore, KD is better suited to stable tasks. In changing environments, it should be combined with re-distillation and drift monitoring.

Table 3. Representative KD studies in Cloud–Edge collaborative intelligence.

Reference	Research Objective	Methodology/ Framework	Optimization Target	Relation to Cloud–Edge and Large–Small Models	Key Contributions and Limitations
Luo et al. [31]	Real-time UAV visual assistance under edge limits	KeepEdge: cloud-to-edge teacher–student distillation with semantic priors	Lower latency and energy with retained accuracy	A cloud teacher guides lightweight edge students	Improves efficiency and accuracy, but needs aligned features.
Chen et al. [32]	Efficient multimodal inference in cloud–edge systems	Selective feature distillation with a multi-branch teacher	Accuracy–latency balance through adaptive paths	Maps multimodal cloud knowledge to unimodal edge encoders	Supports multimodal compression, but may not scale to unseen modalities.
Zhang et al. [33]	Feature transfer for lightweight object detectors	Attention-guided and non-local distillation across feature maps	Higher detection accuracy with moderate overhead	Transfers multi-level knowledge between cloud and edge models	Improves COCO mAP, but depends on temperature and layer alignment.
Qiao et al. [34]	Consistency in asynchronous federated learning	ASMAFL with staleness-aware aggregation and re-distillation	Less drift under non-IID data	Uses hierarchical KD across cloud, edge, and clients	Improves stability, but adds communication in re-distillation rounds.
Li et al. [35]	Dynamic partitioning for hierarchical AI	Real-time adaptive partitioning with changing teacher–student roles	Latency-aware resource allocation and synchronization	Adjusts roles under changing networks	Improves response speed, but needs accurate bandwidth estimates.
Guo et al. [36]	Cross-modal alignment in heterogeneous federated learning	MASA with modality-asymmetric local distillation	Better collaboration across heterogeneous modalities	Bridges strong and weak modality clients	Improves multimodal generalization, but may leak modality mappings.

Energy-efficiency note: Useful sustainability metrics include Joules-per-inference, energy–delay product, communication energy, update energy, and carbon-aware resource use. These metrics matter when distillation reduces cloud computation but adds edge inference, synchronization, or re-distillation cost.

3.2. Split Inference, Feature Compression, and Collaborative Computation

Split inference divides model execution between the cloud and the edge. This allows large models to support edge devices without full local deployment. The main challenge is to balance latency, bandwidth, energy, and accuracy.

Early studies focus on where to split the model. Li et al. [39] use real-time profiling to choose layer-wise partitions. Their method considers computation and wireless conditions. Liang et al. [40] select layer cut points for co-inference. This helps balance computation and communication. However, frequent re-partitioning can add overhead. It can also reduce portability across architectures.

Recent studies focus more on intermediate features. Instead of sending raw data, Wang et al. [41] compress feature maps. They remove spatial and statistical redundancy. This reduces bandwidth while keeping accuracy. Yuan et al. [42] further use scalable feature compression. Their method adapts to changing accuracy and latency needs. These studies turn split inference into semantic-level collaboration.

Dynamic offloading is also important. Zheng et al. [43] jointly optimize model partitioning and transmission. Their method targets semantic communication-assisted MEC systems. Hybrid resource management methods [44] use elastic cloud bursting to relieve edge bottlenecks. These studies show that split inference must consider computation, communication, and semantics together.

Efficient split inference depends on how intermediate representations are handled. Feature maps can be large and redundant. They may include spatial, channel-wise, or temporal redundancy. Feature compression [45] can reduce this cost. It may prune uninformative activations, use bottleneck encoders, quantize tensors, or transmit task-relevant features. This changes split inference from simple partitioning to joint optimization. The system must consider model structure, feature size, communication rate, privacy risk [46], and accuracy. In large–small collaboration, the cloud can provide semantic priors. The edge can perform early perception. It can transmit compact features only when local confidence is low.

Discussion and design implications. Split inference is useful when edge devices cannot run the full model. It also requires reasonably stable cloud–edge links. Its performance depends on partition points [47], feature size, bandwidth change, and privacy risk. It fits latency-sensitive tasks with predictable networks. In bandwidth-limited or privacy-sensitive settings, it needs feature compression, encryption, or local fallback models.

3.3. Quantization, Pruning, and Lightweight Deployment

Quantization and pruning make edge deployment more practical. They reduce memory, computation, bandwidth, and energy cost. Unlike KD, they do not mainly transfer knowledge. They directly make model execution lighter.

Quantization [48,49] maps weights, activations, or features to low-precision formats. Common formats include INT8 and INT4. Mixed precision can also balance efficiency and accuracy. Post-training quantization is useful

when retraining data are limited. Quantization-aware training usually better preserves accuracy. In cloud–edge collaboration, quantization can be applied to edge models, transmitted features, adapter modules, and expert activations. Thus, it is closely linked to split inference and bandwidth-aware coordination.

Pruning removes redundant parts of a model. These parts may include weights, channels, attention heads, tokens, or experts. Structured pruning [50] removes whole filters, channels, or blocks. It is usually easier to accelerate on hardware. Unstructured pruning may reach higher sparsity. But it needs sparse computation support. For edge deployment [51], pruning is often combined with distillation. A cloud-side teacher can help the pruned student recover accuracy.

From a collaboration view, compression defines what edge models can actually run. It affects whether distillation, split inference, federated adaptation, or expert routing is feasible on real hardware.

Discussion and design implications. Quantization, pruning, and lightweight deployment reduce resource use. But aggressive compression [49] can harm robustness, calibration, and generalization. These methods are most useful for constrained devices. They should be selected according to latency, memory, energy, hardware support, and acceptable accuracy loss.

3.4. Federated and Continual Adaptation

This subsection differs from the distillation discussion in Section 3.1. It focuses on long-term distributed adaptation, personalization, drift control, and lifecycle-aware updates across cloud–edge clients.

Federated and continual adaptation extend collaboration beyond one-time knowledge transfer. They support learning across heterogeneous and changing edge environments. Federated distillation replaces gradient aggregation with semantic exchange. Jin et al. [52] propose pFedSD. It lets edge models learn from local historical versions and global teachers. Wu et al. [53] propose FedICT. It supports heterogeneous models through bidirectional feature exchange. Yao et al. [54] propose FedGKD. It reduces client drift by distilling from historical global models. These methods improve scalability and robustness. However, many still need auxiliary data for calibration.

Continual federated learning addresses non-stationary data. Yang et al. [55] use privacy-preserving knowledge fusion. This helps reduce catastrophic forgetting. Zhang et al. [56] study resource-aware multi-task federated adaptation. Their method handles incomplete and evolving data. These methods improve adaptability. But replay and fusion modules add memory and computation cost.

Personalized federated learning balances global consistency and local specialization. Tan et al. [57] group PFL methods into meta-learning, regularization, clustering, and mixture-of-experts approaches. Many methods use a shared global backbone with local adapters. This allows cloud models to act as semantic priors. It also lets edge models adapt to local contexts. However, personalization often assumes stable connectivity. It can also be costly for low-power devices.

Federated and continual adaptation make collaboration more persistent. They move the focus from one-time transfer to long-term semantic alignment. They also support task-aware specialization. Still, lifecycle-aware synchronization and cross-tier governance remain open problems.

Discussion and design implications. Federated and continual adaptation [58] fit privacy-sensitive and non-stationary edge settings. They help when local data cannot be collected centrally. They support personalization and long-term knowledge evolution. But they face non-IID data, client drift, unstable communication, and forgetting [59]. Compared with KD, they capture local changes better. However, they need stronger synchronization, drift detection, and lightweight updates.

3.5. Hybrid Expert and Modular Design

Hybrid expert and modular designs support scalable collaboration. They activate only task-relevant parameters. They also update small modules instead of full models. This helps large–small systems adapt under heterogeneous cloud–edge resources.

Sparse expert routing. Mixture-of-Experts (MoE) uses sparse routing. Each input is sent to only a few experts. MoESys [60] optimizes expert placement, routing, and activation in distributed systems. Sparse gating reduces computation while keeping model capacity. In cloud–edge systems, experts can be placed hierarchically. Latency-critical experts can be cached near the edge. Global experts can remain in the cloud. This supports elastic specialization under changing bandwidth and compute budgets.

Parameter-efficient modular adaptation. Adapter tuning inserts small trainable modules into a frozen backbone. This enables fast task or domain adaptation. It also reduces communication cost. Feng et al. [61] show that adapter-based selective distillation can work with few trainable parameters. This makes it useful for on-device personalization and continual updates.

Low-rank modular fine-tuning. LoRA-style modules separate heavy backbones from lightweight updates. Wang et al. [27] propose federated fine-tuning with LoRA modules. Devices update the LoRA modules. The transformer trunk stays on the server. This reduces uplink traffic and training energy. It also helps maintain stable convergence. Rank- and layer-selective updates enable resource-aware modular coordination.

As shown in Figure 2, MoE, adapters, and LoRA move collaboration away from monolithic deployment. They support sparse activation and modular updating. This provides a practical basis for specialization and continual adaptation.

Hybrid expert and modular designs [62] also support dynamic large–small collaboration. The cloud can maintain global experts, task routers, and semantic priors. Edge nodes can cache or activate lightweight experts for local tasks and resources. Adapter and LoRA modules [63] allow edge devices to update task-specific parameters only. This reduces communication during personalization and continual learning.

These benefits depend on good routing and expert placement. They also need lifecycle management. Without orchestration, too many experts may appear. Module versions may become inconsistent. Local specialization may also become biased. Thus, modular designs should be paired with cloud–edge orchestration policies. These policies should monitor expert use, merge redundant modules, and decide when to update, offload, or retire local experts.

Discussion and design implications. Hybrid expert and modular designs are useful when tasks and resources vary widely. They support selective activation and lightweight updates. But they need orchestration to track experts, merge modules, and avoid model fragmentation.

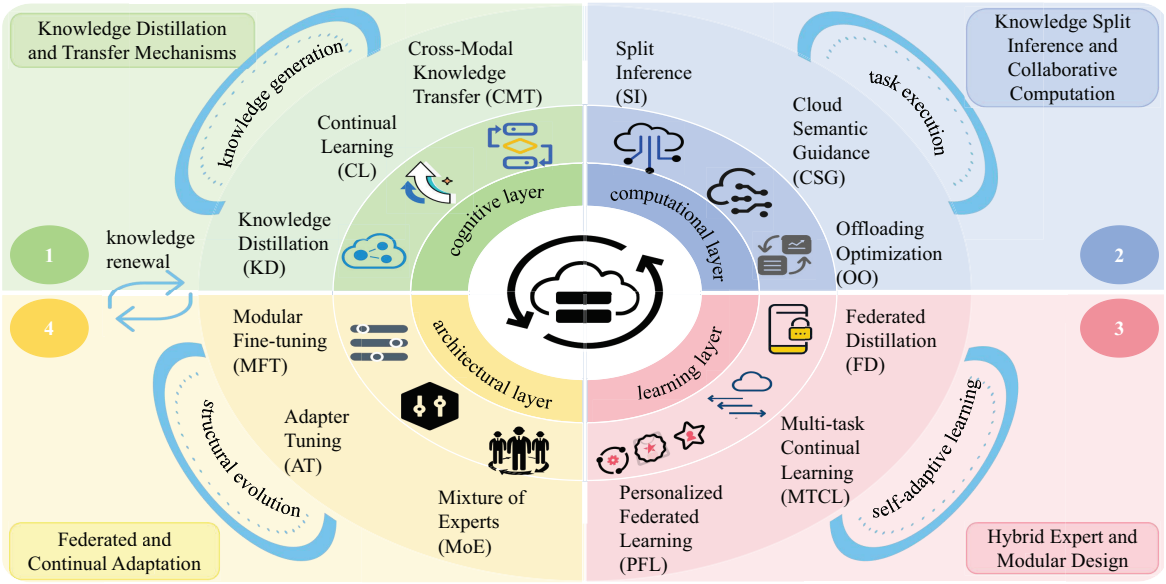

Figure 2. Cloud–Edge collaborative framework. The numbered sectors denote: (1) knowledge distillation and transfer mechanisms, (2) knowledge split inference and collaborative computation, (3) hybrid expert and modular design, and (4) federated and continual adaptation. Quantization, pruning, and lightweight deployment support resource-constrained edge execution. The central loop shows feedback across layers for dynamic alignment in distributed AI systems.

4. Cloud–Edge Collaborative Intelligence Framework

This section links collaboration mechanisms with practical deployment constraints [64,65]. The framework is not only a list of techniques. It can be read as a design process. First, the system identifies the main constraint. Then, it assigns cloud-side and edge-side roles. Next, it selects a suitable collaboration strategy. Finally, it evaluates trade-offs in latency, bandwidth, privacy, energy, adaptability, and robustness.

Different constraints lead to different choices. Latency-sensitive systems [64] often need edge-first inference, split inference, caching, or early exits. Bandwidth-limited systems benefit from distillation, feature compression, quantization, and prototype or logit exchange. Privacy-sensitive systems require federated learning, federated distillation, differential privacy, or secure aggregation. Weak edge hardware favors pruning, low-bit quantization, adapters, LoRA-style tuning, or TinyML-style triggers. Non-stationary environments need drift-aware synchronization, re-distillation, and rollback. Energy- or carbon-constrained systems [66] benefit from sparse expert routing, adaptive offloading, and energy-aware scheduling. Thus, the collaboration strategy should follow the dominant deployment constraint.

To make the proposed framework more actionable, Table 4 summarizes a deployment-oriented decision matrix [64,65] for selecting large–small model collaboration strategies under different cloud–edge constraints, based

on the above mechanisms and recent studies on on-device AI and edge large language models . Instead of treating collaboration as a fixed architecture, the matrix highlights how latency, bandwidth, privacy, device capability, data drift, energy consumption, and safety requirements affect the choice of collaboration mechanism.

Table 4. Deployment-oriented decision matrix for large–small model collaboration in cloud–edge systems.

Dominant Constraint	Preferred Collaboration Strategy	Avoided Risk or Limitation	Representative Metrics
Strict latency requirement	Edge-first inference, early-exit models, local small-model execution, and split inference with shallow cloud interaction	Frequent cloud queries may cause long-tail delay and unstable response under network congestion	Average latency, P_{95}/P_{99} latency, deadline miss ratio
Limited bandwidth	Knowledge distillation, feature compression, quantized intermediate representation transfer, and event-triggered synchronization	Uploading raw data or high-dimensional features can increase communication cost and expose sensitive information	MB per inference, uplink/downlink traffic, synchronization frequency
High privacy or compliance requirement	Federated learning, federated distillation, privacy-preserving adaptation, secure aggregation, and on-device filtering	Centralized raw-data collection may violate data-locality, privacy, or regulatory constraints	Privacy budget, data exposure level, aggregation cost, communication rounds
Weak edge hardware capability	INT8/INT4 quantization, pruning, lightweight adapters, TinyML-style models, and cloud-assisted fallback	Deploying full-size models on edge devices may exceed memory, power, or thermal limits	Model size, peak memory, FLOPs, energy per inference, device utilization
Non-stationary local data	Drift detection, continual adaptation, periodic or event-triggered re-distillation, model versioning, and rollback	Static distillation may lead to outdated edge models and semantic inconsistency with the cloud model	Drift score, adaptation frequency, accuracy degradation, recovery time
Energy and carbon constraint	Sparse expert activation, adaptive distillation frequency, energy-aware offloading, carbon-aware scheduling, and local caching	Redundant cloud invocation, unnecessary synchronization, or always-on large-model inference may increase energy use	Joules per inference, energy–delay product, carbon cost, cloud invocation ratio
Safety-critical or reliability-sensitive tasks	Hybrid local–cloud verification, fail-safe edge models, human-in-the-loop validation, and conservative offloading policies	Unstable offloading or unverified cloud reasoning may reduce robustness in safety-critical scenarios	Failure rate, rollback frequency, confidence score, safety violation rate

This matrix also shows that no single collaboration mechanism is optimal for all deployment scenarios. For latency-sensitive tasks, edge-first execution and early-exit inference are often preferred. For privacy-sensitive tasks, federated learning and federated distillation are more suitable. For bandwidth- or energy-constrained systems, compressed knowledge transfer, sparse expert activation, and adaptive synchronization become more important. Therefore, practical cloud–edge collaboration should be designed as a constraint-aware strategy selection problem rather than a one-size-fits-all architecture.

4.1. Hierarchical Model Orchestration

Hierarchical model orchestration controls how large and small models work across cloud, edge, and end devices. It decides where models run, when they are updated, and how feedback moves across tiers. In general, the cloud handles global reasoning and long-term planning. The edge supports near-real-time coordination and model fusion. End devices perform low-latency sensing, inference, and actuation.

Recent studies connect orchestration with resource control and model coordination. Wang et al. [44] use hybrid cloud scheduling with elastic edge coordination. Their method reduces tail latency under changing workloads. Li et al. [67] use digital twins for industrial IoT orchestration. This supports proactive decisions and event-triggered management across cloud, edge, and end layers. Du et al. [68] model SDN-based orchestration as a Stackelberg game. This balances global optimization and local response. Shokouhi et al. [69] study mobility-aware offloading for service continuity. Cohen et al. [70] optimize service chaining and compute placement across the continuum.

These works show a clear shift in orchestration. The focus moves from static scheduling to feedback-driven control. Practical designs need slow global planning and fast local adaptation. They also need clear interfaces for policy delivery and telemetry feedback. Each tier should keep enough autonomy, but global consistency must be maintained. Observability, rollback, and recovery are also required. These elements form the control basis for scalable cloud–edge collaboration.

4.2. Knowledge Flow and Lifecycle Management

Knowledge flow describes how semantic information moves across cloud and edge. Lifecycle management describes how this knowledge changes over time. The main challenge is to keep global models aligned with local changes. This is difficult in heterogeneous and non-stationary environments.

Early cloud–edge systems often use downward knowledge transfer. A cloud teacher distills compact knowledge to edge students. KeepEdge [31] and selective feature distillation [32] show this idea. They reduce bandwidth use and speed up edge inference. But one-way transfer has a clear limit. It cannot fully use local feedback or handle local distribution shifts.

Recent methods use bidirectional and feedback-driven knowledge flow. Federated Co-Distillation (FCD) [71] supports semantic exchange through soft-target distillation. It models upward knowledge flow from the edge to the cloud. Efficient Parallel Split Learning (EPSL) [72] uses activation and gradient exchange. Cost-aware partitioning [73] links knowledge exchange with resource optimization. These methods turn static distillation into continuous cross-tier learning.

Knowledge flow also needs measurable lifecycle control. First, knowledge transfer can be measured by teacher–student performance gap, distillation loss, representation similarity, feature alignment score, calibration error, and transfer efficiency. Transfer efficiency can be measured per unit of communication or energy cost. Second, lifecycle-aware collaboration [74] should monitor data drift, concept drift, and representation drift [75]. These can be tracked through prediction confidence, entropy, embedding statistics, or local–global representation divergence. Third, unstable networks require event-triggered re-distillation [76], asynchronous updates, partial parameter transfer, version control, and rollback. Together, these steps form a closed loop. The loop includes edge sensing, feedback collection, cloud update, and edge adaptation.

Neuro-symbolic and control-theoretic views can make this loop safer [77,78]. Symbolic rules, task constraints, safety policies, and domain knowledge can guide cloud-side reasoning. They can also check whether edge execution matches the global semantic intent. From a control view, dynamic offloading should be stability-aware. Event-triggered updates, hysteresis thresholds, bounded update frequency, uncertainty estimation, and fail-safe rollback can reduce oscillation. They also reduce delayed updates and inconsistent model states under changing bandwidth, workload, and device conditions.

Lifecycle management is also needed after deployment. Digital-twin orchestration [67] can virtualize system states. It can guide model refresh, synchronization, and rollback. MLOps pipelines [79] provide monitoring, drift detection, and automated redeployment. These tools move knowledge flow from training-time exchange to deployment-time governance. Still, open problems remain. These include adaptive synchronization, privacy-preserving upward transfer, and semantic consistency under changing conditions.

4.3. Resource Scheduling and Elastic Offloading

Resource scheduling decides where, when, and how much computation should run. Elastic offloading makes this decision adjustable over time. In large–small collaboration, scheduling must consider computation, communication, energy, and model accuracy together.

Early studies often model offloading as an energy–latency trade-off. Wang et al. [80] optimize split ratio, transmission power, and CPU frequency. Their method targets vehicular edge systems under delay limits. Zheng et al. [43] study DNN partitioning in semantic communication-assisted MEC. They jointly optimize model splitting and wireless allocation. Younis et al. [81] use approximate computing in convex optimization. This gives guarantees for energy–accuracy trade-offs. Bi et al. [82] study joint offloading and user association. Their work shows the effect of cost and topology constraints.

Learning-based schedulers address more dynamic settings. Fan et al. [83] design queue-aware offloading and power control for cooperative MEC. Yang et al. [84] use multi-objective deep reinforcement learning. Their method optimizes energy and latency under changing network conditions. Shao et al. [85] study task-oriented communication. They use the information bottleneck principle to learn compact features. This connects model partitioning with communication efficiency.

Table 5 summarizes representative studies. These works show that cloud–edge collaboration needs joint optimization of computation, communication, and semantics. However, many methods still treat scheduling, adaptation, and semantic alignment separately. A unified control mechanism remains needed.

Table 5. Representative studies on resource scheduling and elastic offloading in Cloud–Edge collaborative intelligence.

Reference	Research Objective	Methodology/ Framework	Optimization Target	Relation to Cloud–Edge	Key Contributions and Limitations
Yuan et al. [73]	Cost-aware hybrid offloading	Partial and cost-minimized computation partitioning	Total cost and latency reduction	Adds economic cost modeling to offloading design	Provides a cost-efficient baseline for hybrid scheduling.
Wang et al. [80]	Energy-minimized partial offloading	Joint optimization of split ratio, power, and CPU frequency	Energy use under delay constraints	Cloud-assisted vehicular edge system	Shows dynamic energy–delay balance; focuses on vehicular settings.
Zheng et al. [43]	Computation-aware inference offloading	Semantic communication-assisted DNN partitioning	Latency reduction with retained accuracy	Cloud–edge hierarchical DNN inference	Uses semantic compression to improve communication efficiency.
Younis et al. [81]	Approximate computing-based offloading	Convex optimization for energy–latency trade-off	Power efficiency with bounded accuracy loss	Energy-limited edge nodes	Provides a mathematical formulation; lacks dynamic workload adaptation.
Bi et al. [82]	Cost-minimized offloading and association	Multi-user hybrid optimization of offloading and association	Global cost minimization and fairness	Hybrid cloud–edge architecture	Links user association with task offloading and improves resource balance.
Fan et al. [83]	Cooperative MEC resource control	Queue-aware scheduling and power allocation	Delay reduction and throughput improvement	Multi-agent cooperative edge nodes	Achieves bounded latency through dynamic scheduling; uses single-layer coordination.
Yang et al. [84]	Multi-objective resource scheduling	PPO-based deep reinforcement learning framework	Joint energy and latency reduction	Task scheduling in MEC architecture	Learns adaptive energy–delay trade-offs; has high training overhead.
Shao et al. [85]	Task-oriented communication	Information Bottleneck-based encoding	Rate–accuracy trade-off	Semantic-aware edge inference framework	Connects computation and communication; assumes stable channels.

4.4. Trustworthy and Privacy-Aware Collaboration

Trustworthy collaboration protects knowledge exchange across cloud and edge. In large–small systems, models exchange features, gradients, predictions, and updates. These data must be protected during transmission, training, and execution.

Secure aggregation protects local updates in federated learning. Wang et al. [86] propose VOSA. It verifies updates without exposing gradients. Zhu et al. [87] design malicious-resistant non-interactive aggregation. It reduces forgery and collusion risks. These methods protect communication between edge clients and the cloud.

Differential privacy protects sensitive information during learning. Zhou et al. [88] study adaptive iteration control for differentially private FL. Their method balances convergence and privacy budgets. Ma et al. [89] use sparse perturbation to reduce cumulative noise. This is useful in bandwidth-limited settings. These methods improve privacy while keeping learning quality.

Secure execution protects data during inference. Liu et al. [90] combine trusted execution environments with verifiable inference outsourcing. This protects both confidentiality and correctness. Kim and Guyot [91] use fully homomorphic encryption for CNN inference. This gives formal privacy guarantees without trusted hardware.

These methods form a layered defense. Secure aggregation protects transmission. Differential privacy protects learning. Trusted execution and homomorphic encryption protect inference. Still, most methods handle security separately. A unified control framework is still needed. It should connect privacy budgets, secure scheduling, and semantic consistency across tiers.

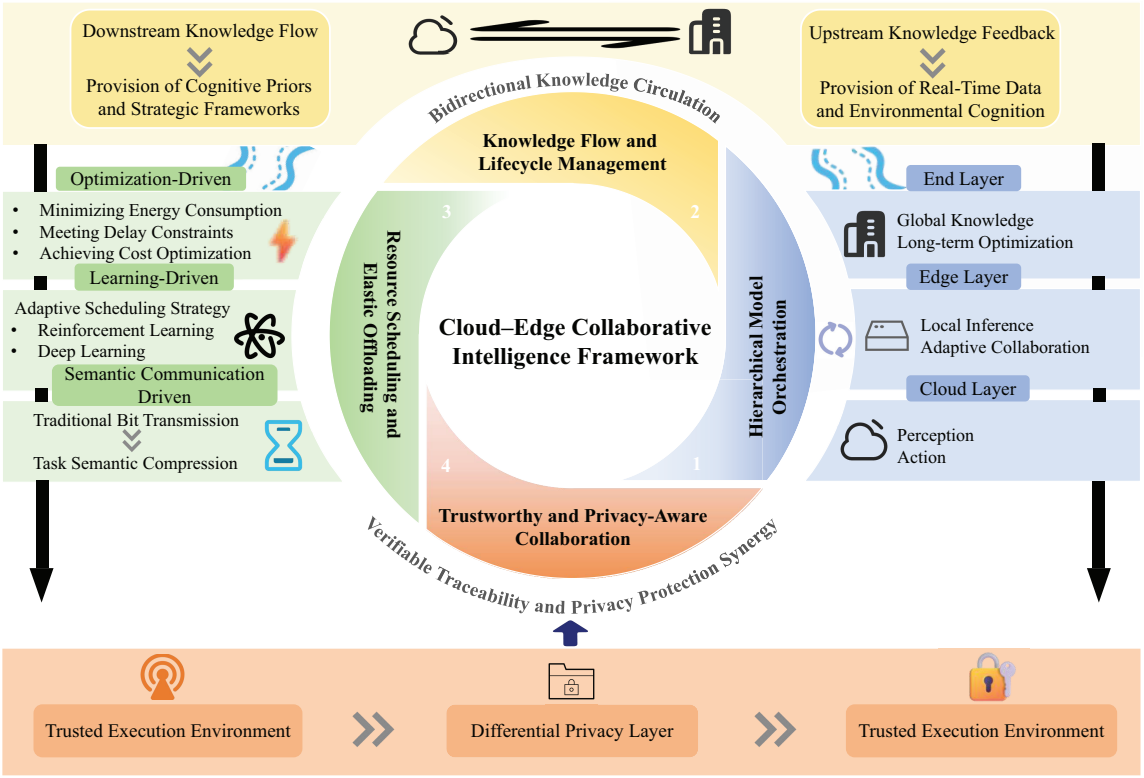

Figure 3. Cloud–Edge collaborative intelligence framework. The figure shows hierarchical model orchestration, knowledge flow and lifecycle management, resource scheduling and elastic offloading, and privacy-aware collaboration across cloud, edge, and end layers. It also shows how control, knowledge, data, and security planes support adaptive large–small model collaboration.

5. Agent-Enhanced Collaboration and Autonomy

5.1. Agent-Orchestrated Large–Small Model Systems

Agent-based orchestration adds an autonomous control layer to cloud–edge collaboration. It helps coordinate cloud-side large models and edge-side small models under changing workloads. Reinforcement learning (RL) and multi-agent RL (MARL) can support this process. Agents can learn where to place computation, how to partition models, and when to offload tasks. This helps align system-level goals with local edge constraints.

Single-agent orchestration. Single-agent methods are useful when centralized control is still feasible. Yamansavascilar et al. [92] propose DeepEdge. It is based on a Double Deep Q-Network (DDQN). The agent learns offloading and routing policies in mobile edge networks. It interacts with workload arrivals and channel changes. This allows it to replace fixed heuristic schedulers. It also supports context-aware latency optimization. Such designs fit small- or medium-scale MEC systems. However, they mainly optimize resources. They rarely include model-level knowledge transfer or semantic feedback.

Multi-agent coordination. Multi-agent methods are better suited to large and heterogeneous systems. Li et al. [93] propose a scheduled MARL framework. Multiple agents cooperate to manage task queues and resources. Hsieh et al. [94] develop decentralized cooperative control for heterogeneous MEC systems. Their method improves load balancing and service continuity. MARL improves scalability and resilience. Yet it also introduces agent communication cost. It often depends on carefully designed rewards. This can weaken interpretability and stability under sudden workload changes.

Hierarchical agent architectures. Hierarchical agents better match the cloud–edge structure. Sun et al. [95] propose a hierarchical deep reinforcement learning framework. It separates high-level planning from low-level execution. High-level planning includes service caching and global offloading. Low-level execution includes edge-side scheduling. This structure fits cloud-side reasoning and edge-side execution. However, hierarchical RL adds nested optimization loops. These loops may cause unstable convergence. Few studies include semantic knowledge flow or adaptive re-distillation in agent decisions.

Agent-based orchestration is useful for adaptive cloud–edge control. Still, many existing studies remain computation-centered. They do not fully connect agents with distillation, semantic alignment, or model updates. They also face scalability, interpretability, and robustness issues. Future agent frameworks should be model-aware and verifiable. They should jointly manage scheduling, model adaptation, and semantic consistency.

From a deployment view, single-agent and multi-agent orchestration [96] fit different settings. Single-agent methods are easier to train and analyze. They are suitable for small- or medium-scale MEC systems. This holds when centralized observation and decision-making remain practical. However, they may not scale well. The problem becomes harder as edge nodes, tasks, and models increase.

Multi-agent methods fit larger and more heterogeneous cloud–edge systems. Local agents can make distributed decisions. They can also negotiate resource allocation. However, they add communication overhead [97]. They also face training non-stationarity, reward-design difficulty, and stability risks. Therefore, agent-based orchestration should be viewed as a coordination layer. It should not fully replace conventional scheduling. Its use requires care in scalability, interpretability, and robustness.

5.2. Multi-Agent Collaboration and Negotiation

Multi-agent systems provide a natural way to support plan–execute–reflect loops. In cloud–edge collaboration, cloud agents can plan at a higher level. Edge agents can execute decisions under local constraints. Reflection closes the loop. It updates policies and roles across tiers based on feedback.

Several studies show how agents can negotiate through system signals. Nguyen et al. [98] propose a blockchain-empowered MADRL framework. Agents jointly optimize task offloading and consensus operations. Reputation and consensus act as negotiation signals. They help agents adapt policies based on trust and performance. However, blockchain feedback adds latency. This limits its use in real-time edge scenarios.

Huang et al. [99] use a dual-timescale control model. It separates slow planning from fast execution. This structure fits the plan–execute–reflect loop. It also improves stability under workload changes. Yet its negotiation is still resource-centered. It does not model semantic interaction between large and small models.

Other studies use shared states, attention, and security feedback. Liu et al. [100] apply MARL to SCMA-aided IoT edge systems. Agents evolve policies through shared states and adaptive exploration. Gao et al. [101] use attention-weighted recurrent MARL. Attention scores work as soft negotiation signals. Li et al. [102] design adversarial MARL for UAV-enabled edge systems. Security feedback is included in cooperative learning. These methods improve cooperation. Still, reflection often focuses on convergence or robustness. It rarely supports semantic alignment.

Multi-agent collaboration supports distributed planning and feedback-driven adaptation. However, most frameworks still optimize computation or communication. They rarely use semantic priors from cloud-side large models. They also seldom connect reflection with distillation, drift detection, or hierarchical alignment. Future work should develop model-aware multi-agent negotiation. Cloud agents can provide semantic constraints. Edge agents can refine decisions with lightweight models. Runtime reflection should include telemetry, semantic checks, and safety verification. This would make agent cooperation more reliable.

5.3. Cognitive and Semantic Synergy

Cognitive and semantic synergy connects cloud reasoning with edge perception. In large–small collaboration, agents can serve as the bridge. Cloud models provide semantic priors and abstract knowledge. Edge models provide local observations and environmental feedback. This interaction links perception, reasoning, and decision-making. It supports adaptive autonomy under changing networks and resources.

Several studies move toward this goal through expert selection and modular adaptation. Lin et al. [103] propose dynamic expert pruning and routing for MoE inference. The method activates only task-relevant experts. This reduces memory and computation on constrained devices. However, it lacks cross-node coordination. It also lacks semantic-aware routing across the cloud–edge continuum.

Kong et al. [104] develop a system framework for serving MoE models at the edge. It optimizes expert caching, activation, and offloading. This improves computation–communication trade-offs. However, adaptation is mainly driven by hardware conditions. Agent supervision could add semantic relevance and long-term intent. This would move expert placement beyond system metrics.

Guo et al. [105] use MoE modules as lightweight adapters. This supports continual vision–language adaptation with low parameter overhead. Yet uncontrolled expert growth may create redundancy and delay. Zhang et al. [106] propose ACL for edge–cloud collaborative learning. It partitions tasks and tunes resources for large language model services. ACL improves latency and energy trade-offs. Still, it lacks a unified semantic interface between reasoning and runtime orchestration.

Current studies improve expert routing, modular adaptation, and system scheduling. But a full cognitive loop is still missing. Such a loop should connect cloud semantic intent, edge feedback, and agent decisions. Future

systems should let agents manage expert routing and adapter activation. They should also control cloud–edge synchronization under semantic uncertainty. This is needed for interpretable and adaptive large–small collaboration.

5.4. Towards Self-Evolving Collaborative Ecosystems

Self-evolving ecosystems are the next step for cloud–edge collaboration. Such systems should adapt computation, models, and policies over time. They must work under heterogeneous and non-stationary conditions. Meta-learning and adaptive scheduling are useful tools for this goal.

Temporal adaptation is one important direction. Hao et al. [107] propose a multi-timescale deep reinforcement learning framework. It separates fast edge reactions from slower cloud optimization. This helps stabilize performance under dynamic workloads. However, the learning models are still treated as fixed components. Their representations do not evolve with the system.

Meta-learning improves transfer across tasks and environments. Sharma et al. [108] propose deep meta Q-learning for multi-task offloading. Agents use meta-initialization to generalize across heterogeneous settings. This speeds up convergence and task transfer. However, deterministic adaptation may be less robust under uncertainty.

Uncertainty-aware learning is also important. Wang and Wong [109] use Bayesian meta-learning for scheduling and prediction. The method quantifies uncertainty in volatile wireless settings. This supports risk-aware decisions. However, it focuses mainly on prediction tasks. It does not fully connect with hierarchical cloud–edge orchestration.

Lifelong and structure-aware learning further extend this direction. Dai et al. [110] study lifelong meta-reinforcement learning in vehicular edge systems. Their method supports policy transfer with limited retraining. Yet synchronization between local adaptation and cloud optimization remains underexplored. Liu et al. [111] combine graph neural networks with reinforcement learning. Their method models task dependencies in dynamic vehicular clouds. This supports semantic-aware scheduling. Still, it has limited integration with meta-adaptation and uncertainty reasoning.

These studies show a path from reactive scheduling to adaptive learning. They also introduce uncertainty-aware and structure-aware decisions. But a unified cognitive–control architecture is still lacking. Future systems should link temporal coordination, meta-learning, uncertainty reasoning, and structural awareness. They should also manage expert routing, adapter lifecycles, and cloud–edge synchronization. A continual prune–merge–distill process could support this goal. The overall framework is shown in Figure 4.

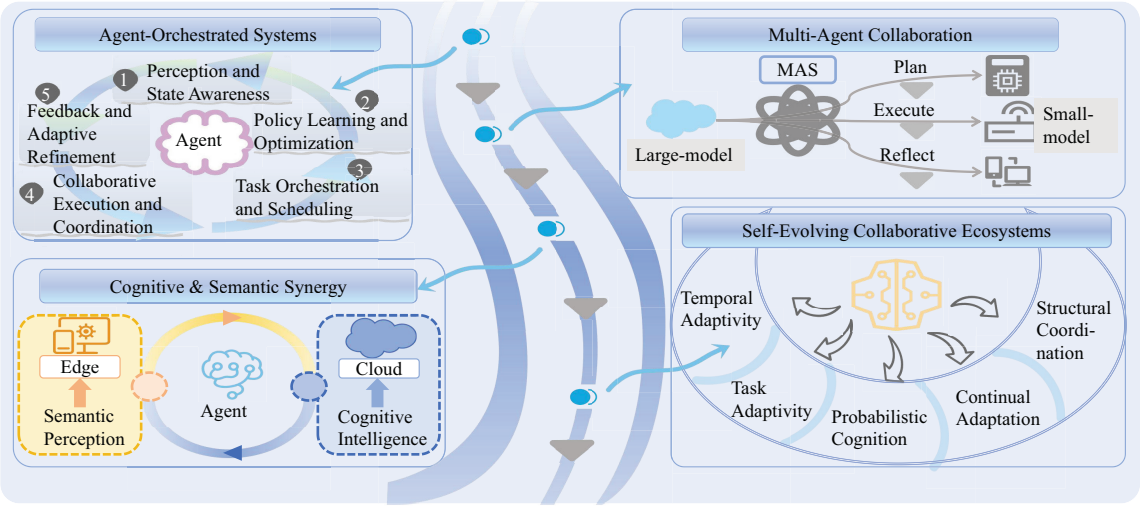

Figure 4. Agent-enhanced collaboration and autonomy. The figure shows single-agent orchestration, multi-agent negotiation, cognitive and semantic synergy, and self-evolving feedback loops. These components coordinate cloud-side large models and edge-side small models. The plan–execute–reflect process supports adaptive scheduling, semantic alignment, and autonomous cloud–edge collaboration.

6. Open Challenges and Future Directions

Large–small model collaboration has made clear progress in cloud–edge environments. Still, a unified and self-sustaining ecosystem is not yet available. Current systems often solve separate problems, such as offloading, compression, or privacy. They rarely address knowledge alignment, autonomy, trust, and sustainability together. This section discusses the main open challenges.

6.1. Dynamic Knowledge Transfer and Unified Multimodal Representation

Dynamic knowledge transfer remains a core challenge. Current methods, such as distillation, split learning, and federated adaptation [112], often assume fixed model structures. They also rely on relatively stable representation spaces. Real-world cloud–edge systems are different. They involve changing tasks, heterogeneous devices, and multimodal data. Large cloud models should update edge models based on uncertainty, drift, and local feedback. The system must decide what to transfer, when to synchronize, and how to align representations. These decisions must also respect bandwidth and privacy limits.

Multimodal foundation models make this problem harder [113]. Cloud–edge systems may need to align vision, language, sensing, and control signals. A shared latent space is difficult to build across different devices and tasks. Future work should study modality-aware distillation. It should also improve representation alignment and cross-modal compression. These methods are needed for stable knowledge evolution.

Real deployments also need scenario-specific validation. In autonomous driving [114], cloud-side large models can support scene reasoning and long-term planning. They can also analyze rare events. Edge-side small models handle object detection, tracking, localization, and safety control. In industrial IoT [115], cloud models can combine sensor streams, maintenance logs, and production records. They support global fault diagnosis and process optimization. Edge models provide fast anomaly detection and local control. In smart healthcare [116], large multimodal models can combine medical images, records, and physiological signals. Edge models on wearable or bedside devices support monitoring, alerts, and local inference.

Future deployment studies should evaluate more than accuracy. They should report latency, bandwidth use, memory footprint, energy use, and update frequency. They should also test failure recovery and robustness under changing networks. Missing modalities, asynchronous sensing, and cross-modal distillation should also be considered.

6.2. Autonomous Cloud–Edge–End Scheduling and Ecosystem-Level Coordination

Autonomous scheduling must move beyond single offloading decisions. Reinforcement learning and optimization methods can improve resource allocation. However, many current systems are still reactive and task-centered. Future systems should support self-regulating cloud–edge governance.

Future frameworks should combine multi-agent coordination, digital twins, and predictive control [117]. This can support cross-layer scheduling across cloud, edge, and end devices. Key tasks include expert placement, workload prediction, and policy negotiation. Scheduling should not optimize only latency or energy. It should also consider long-term fairness, stability, and system utility. Local decisions must remain flexible. At the same time, they should follow cloud-level semantic intent and global objectives.

6.3. Trustworthy Collaboration and Incentive Mechanisms

Trust is still a major barrier for large–small model collaboration. Secure aggregation, differential privacy, and trusted execution environments [118] protect specific stages. Yet they do not form a complete trust framework. Future systems need protection across training, inference, and lifecycle management.

Verifiable and incentive-compatible collaboration is needed. This includes privacy-budget control across tiers. It also includes secure expert migration and defense against adversarial agents. Incentive mechanisms may also be useful. Blockchain or token-based schemes could encourage cooperation among agents and infrastructure providers. However, these mechanisms should not add too much delay. A practical trust framework must balance transparency, accountability, and efficiency.

6.4. Human-in-the-Loop and Human-Agent Collaboration

Human feedback is an important signal for cloud–edge collaboration [119]. Edge devices often interact directly with users, operators, clinicians, drivers, or engineers. This makes them suitable for collecting corrections, preferences, annotations, and safety feedback. Such feedback can help edge models adapt to local contexts. It can also help cloud models improve over time.

Privacy must be protected during this process. Federated learning, federated distillation, differential privacy, and secure aggregation can help [120]. These methods can transfer aggregated feedback to cloud-side large models. They can do this without exposing raw user data.

Human-agent collaboration also creates new risks. Human feedback may be sparse, noisy, biased, delayed, or inconsistent [121]. Poorly filtered feedback may amplify bias. It may also reduce global model consistency. Future

systems should assess feedback quality. They should support privacy-aware preference aggregation, incentive design, and human-supervised rollback. Human feedback can make autonomous systems more controllable and accountable.

6.5. Agentic Autonomy, Interpretability, and Ethical Governance

Agent-based orchestration needs stronger interpretability and safety. Multi-agent systems may show unexpected behavior under changing workloads. These behaviors can be hard to predict or verify. This is risky for cloud–edge systems that support real-time decisions.

Future work should develop model-aware and verifiable agent frameworks. Agents should use semantic reasoning, uncertainty estimation, and safety constraints. Formal verification, runtime monitoring, and fail-safe rollback are also needed [122]. These tools can reduce cascading failures. Ethical issues also need attention. Data ownership, bias propagation, model accountability, and environmental cost should be addressed early. These issues affect whether collaborative AI can be deployed responsibly.

6.6. Green AI and Sustainable Deployment

Sustainability is a key challenge for cloud–edge collaboration. Large foundation models require high computation and energy. In collaborative systems, synchronization, expert activation, and retraining can add more cost. Future systems should follow green AI principles [123]. Useful strategies include adaptive precision, expert pruning, energy-aware offloading, and carbon-aware scheduling. Modular fine-tuning and lightweight adaptation can reduce retraining cost. Edge inference can also reduce long-distance data transfer.

Sustainability should be measured at the whole-pipeline level. Accuracy and latency are not enough. Future benchmarks should report energy-oriented metrics [66,124]. These include Joules-per-inference, energy–delay product, communication energy, update energy, and carbon-aware resource use. This is important because collaboration can shift energy cost across tiers. It may reduce cloud computation but increase local inference or communication cost.

Future evaluations should compare centralized cloud inference with cloud–edge collaborative inference under the same task, hardware profile, latency target, and network condition. A simple energy-accounting protocol can decompose the total energy of a collaborative system into edge inference energy, communication energy, cloud inference energy, synchronization energy, and model update energy. Such a protocol makes Joules-per-inference, energy–delay product, and carbon cost more comparable across different collaboration strategies.

Energy-saving strategies should therefore go beyond model compression. They should include sparse expert activation, adaptive distillation frequency, drift-triggered updating, energy-aware offloading, and carbon-aware scheduling [125]. These strategies can make large–small collaboration more sustainable in practice.

Overall, future cloud–edge intelligence should connect knowledge alignment, autonomous control, trust, human feedback, and sustainability. Progress in these areas will move large–small collaboration beyond resource optimization. It will support more reliable, accountable, and energy-aware distributed AI systems.

7. Conclusions

This paper reviewed large–small model collaboration in cloud–edge intelligent systems. We argued that cloud and edge should not be treated as separate computing tiers. They should work as a coordinated system for knowledge sharing, model adaptation, and resource control.

The survey examined several key mechanisms. These include knowledge distillation, split inference, quantization, pruning, lightweight deployment, modular expert design, federated and continual adaptation, agent orchestration, and lifecycle management. Together, these studies show a shift in cloud–edge AI. The focus is moving from simple computation offloading to model-level collaboration and semantic coordination.

Large–small model collaboration still faces open problems. Cloud-side foundation models can provide global semantic knowledge. Edge-side small models can support fast local adaptation. Agents can help coordinate planning, execution, and feedback. However, this vision requires stable orchestration, dynamic knowledge transfer, privacy-aware control, and sustainable deployment.

Cloud–edge collaborative intelligence is therefore a promising direction for adaptive and self-evolving AI systems. Its practical use still requires rigorous benchmarks, reliable orchestration mechanisms, privacy-preserving lifecycle management, and sustainability-aware deployment.

Author Contributions

J.B.: Conceptualization, Methodology, Project administration, Writing—original draft preparation; Y.Y.: Literature search, Classification, Data collection, Visualization; H.Y.: Supervision, Formal analysis, Resources, Validation; Z.W.:

Investigation, Taxonomy refinement, Literature analysis; JL: Comparative study, Table construction, Writing—review & editing; J.Z. (Jiahui Zhai): Investigation, Literature analysis; J.Z. (Jia Zhang): Literature filtering, Reference management, Formatting. All authors have read and agreed to the published version of the manuscript.

Funding

This work was supported by the Beijing Natural Science Foundation under Grants L233005 and 4232049, the National Key Research and Development Program of China under Grant 2025YFE0203700, the National Natural Science Foundation of China under Grants 62473014 and 62173013.

Institutional Review Board Statement

Not applicable.

Informed Consent Statement

Not applicable.

Data Availability Statement

The data will be made available upon reasonable request.

Conflicts of Interest

The authors declare no conflict of interest.

Use of AI and AI-Assisted Technologies

During the preparation of this work, the authors used ChatGPT (GPT-5.5 Thinking, OpenAI) to assist with proofreading, language polishing, and editorial checking. The authors reviewed and edited all AI-assisted outputs and take full responsibility for the content of the published article.

References

1. Yang, W.; Liu, M.; Wang, Z.; et al. Foundation Models Meet Visualizations: Challenges and Opportunities. *Comput. Vis. Media* **2024**, *10*, 399–424.
2. Wang, C.; Peng, Y.; Zhang, D.; et al. A Lightweight Cloud–Edge Collaborative Intelligence Inference Framework with Runtime Dynamic Optimization for Resource-Constrained Consumer Electronics. *IEEE Trans. Consum. Electron.* **2025**, *71*, 6041–6054.
3. An, S.; Jeong, H.; Kim, S.; et al. CINELL: An Energy-Efficient Compute-In/Near-Memory eDRAM Processor for Sparse Transformer-Based Large Language Models. *IEEE Trans. Very Large Scale Integr. Syst.* **2026**, *34*, 242–253.
4. Sun, L.; Tian, J.; Muhammad, G. FedKC: Personalized Federated Learning with Robustness Against Model Poisoning Attacks in the Metaverse for Consumer Health. *IEEE Trans. Consum. Electron.* **2024**, *70*, 5644–5653.
5. Adil, M.; Khan, M.K.; Jadoon, M.M.; et al. An AI-Enabled Hybrid Lightweight Authentication Scheme for Intelligent IoMT Based Cyber-Physical Systems. *IEEE Trans. Netw. Sci. Eng.* **2023**, *10*, 2719–2730.
6. Gu, H.; Zhao, L.; Han, Z.; et al. AI-Enhanced Cloud-Edge-Terminal Collaborative Network: Survey, Applications, and Future Directions. *IEEE Commun. Surv. Tutorials* **2024**, *26*, 1322–1385.
7. Qu, G.; Chen, Q.; Wei, W.; et al. Mobile Edge Intelligence for Large Language Models: A Contemporary Survey. *IEEE Commun. Surv. Tutorials* **2025**, *27*, 3820–3860.
8. Wang, Y.; Yang, C.; Lan, S.; et al. End-Edge-Cloud Collaborative Computing for Deep Learning: A Comprehensive Survey. *IEEE Commun. Surv. Tutorials* **2024**, *26*, 2647–2683.
9. Hoffpauir, K.; Simmons, J.; Schmidt, N.; et al. A Survey on Edge Intelligence and Lightweight Machine Learning Support for Future Applications and Services. *J. Data Inf. Qual.* **2023**, *15*, 20.
10. Kairouz, P.; McMahan, H.B. Advances and Open Problems in Federated Learning. *Found. Trends Mach. Learn.* **2021**, *14*, 1–210.
11. Cheng, Y.; Wang, D.; Zhou, P.; et al. A Survey of Model Compression and Acceleration for Deep Neural Networks. *arXiv* **2017**, arXiv:1710.09282.
12. Alajlan, N.N.; Ibrahim, D.M. TinyML: Enabling of Inference Deep Learning Models on Ultra-Low-Power IoT Edge Devices for AI Applications. *Micromachines* **2022**, *13*, 851.
13. Xu, M.; Cai, D.; Yin, W.; et al. Resource-Efficient Algorithms and Systems of Foundation Models: A Survey. *ACM Comput. Surv.* **2025**, *57*, 110.
14. Song, S.; Li, X.; Li, S.; et al. How to Bridge the Gap between Modalities: Survey on Multimodal Large Language Model. *IEEE Trans. Knowl. Data Eng.* **2025**, *37*, 5311–5329.

15. Wang, X.; Tang, Z.; Guo, J.; et al. Empowering Edge Intelligence: A Comprehensive Survey on On-Device AI Models. *ACM Comput. Surv.* **2025**, *57*, 1–39.
16. Zheng, Y.; Chen, Y.; Qian, B.; et al. A Review on Edge Large Language Models: Design, Execution, and Applications. *ACM Comput. Surv.* **2025**, *57*, 209.
17. Zeng, T.; Zhang, X.; Duan, J.; et al. An Offline-Transfer-Online Framework for Cloud–Edge Collaborative Distributed Reinforcement Learning. *IEEE Trans. Parallel Distrib. Syst.* **2024**, *35*, 720–731.
18. Yan, X.; Li, Z.; Ni, Z.; et al. Traffic-Aware Cloud–Edge Collaborative Offloading for Vehicular Tasks at Complex Intersections. *IEEE Trans. Veh. Technol.* **2026**, *75*, 321–334.
19. Li, X.; Li, H.; Sun, C.; et al. Edge-Enhanced Intelligence: A Comprehensive Survey of Large Language Models and Edge–Cloud Computing Synergy. *IEEE Commun. Surv. Tutorials* **2026**, *28*, 1248–1284.
20. Cheng, L.; Zhang, S.; Zhang, H.; et al. Large-Small Model Collaboration in Mobile Edge Networks with Heterogeneous Computational Resources. *IEEE J. Sel. Areas Commun.* **2026**, *44*, 2733–2749.
21. Rajpal, T.S.; Naithani, A. NeuroCrypt: A Neuro Symbolic AI Ecosystem for Advanced Cryptographic Data Security and Transmission. *IEEE Trans. Artif. Intell.* **2026**, *7*, 512–521.
22. Li, H.; Li, X.; Fan, Q.; et al. Adaptive Model Partitioning and Pruning for Collaborative DNN Inference in Mobile Edge–Cloud Computing Networks. *IEEE Trans. Mob. Comput.* **2026**, *25*, 3744–3759.
23. Lim, J.-A.; Lee, J.; Kwak, J.; et al. Cutting-Edge Inference: Dynamic DNN Model Partitioning and Resource Scaling for Mobile AI. *IEEE Trans. Serv. Comput.* **2024**, *17*, 3300–3316.
24. Bi, J.; Yuan, H.; Duanmu, S.; et al. Energy-Optimized Partial Computation Offloading in Mobile-Edge Computing with Genetic Simulated-Annealing-Based Particle Swarm Optimization. *IEEE Internet Things J.* **2021**, *8*, 3774–3785.
25. Bi, J.; Wang, Z.; Yuan, H.; et al. Cost-Minimized Partial Computation Offloading in Cloud-Assisted Mobile Edge Computing Systems. In Proceedings of the 2023 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Honolulu, HI, USA, 1–4 October 2023; pp. 5052–5057.
26. Zhao, H.; Wang, Z.; Cheng, G.; et al. Online Workload Scheduling for Social Welfare Maximization in the Computing Continuum. *IEEE Trans. Serv. Comput.* **2025**, *18*, 2267–2280.
27. Wang, Z.; Zhou, Y.; Shi, Y.; et al. Federated Fine-Tuning for Pre-Trained Foundation Models Over Wireless Networks. *IEEE Trans. Wirel. Commun.* **2025**, *24*, 3450–3464.
28. Ramadan, M.N.A.; Ali, M.A.H.; Khoo, S.Y.; et al. Federated Learning and TinyML on IoT Edge Devices: Challenges, Advances, and Future Directions. *ICT Express* **2025**, *11*, 754–768.
29. Yang, C.; Zhu, Y.; Lu, W.; et al. Survey on Knowledge Distillation for Large Language Models: Methods, Evaluation, and Application. *ACM Trans. Intell. Syst. Technol.* **2025**, *16*, 143.
30. Shao, J.; Wu, F.; Zhang, X.; et al. Selective Knowledge Sharing for Privacy-Preserving Federated Distillation without a Good Teacher. *Nat. Commun.* **2024**, *15*, 349.
31. Luo, H.; Chen, T.; Li, X.; et al. KeepEdge: A Knowledge Distillation Empowered Edge Intelligence Framework for Visual Assisted Positioning in UAV Delivery. *IEEE Trans. Mob. Comput.* **2023**, *22*, 4729–4741.
32. Chen, J.; Xu, W.; Fan, Y.; et al. Fast Multimodal Edge Inference via Selective Feature Distillation. *IEEE Trans. Mob. Comput.* **2025**, *24*, 11337–11350.
33. Zhang, L.; Ma, K. Structured Knowledge Distillation for Accurate and Efficient Object Detection. *IEEE Trans. Pattern Anal. Mach. Intell.* **2023**, *45*, 15706–15724.
34. Qiao, D.; Guo, S.; Zhao, J.; et al. ASMAFL: Adaptive Staleness-Aware Momentum Asynchronous Federated Learning in Edge Computing. *IEEE Trans. Mob. Comput.* **2025**, *24*, 3390–3406.
35. Li, Y.; Liu, Z.; Kou, Z.; et al. Real-Time Adaptive Partition and Resource Allocation for Multi-User End-Cloud Inference Collaboration in Mobile Environment. *IEEE Trans. Mob. Comput.* **2024**, *23*, 13076–13094.
36. Guo, J.; Fu, Y.; Zhai, Z.; et al. MASA: Multimodal Federated Learning Through Modality-Aware and Secure Aggregation. *IEEE Trans. Mob. Comput.* **2025**, *24*, 7328–7344.
37. Yang, S.; Yang, J.; Zhou, M.; et al. Learning From Human Educational Wisdom: A Student-Centered Knowledge Distillation Method. *IEEE Trans. Pattern Anal. Mach. Intell.* **2024**, *46*, 4188–4205.
38. Zhang, S.; Luo, Y.; Lyu, Z.; et al. ShiftKD: Benchmarking Knowledge Distillation under Distribution Shift. *Neural Netw.* **2025**, *192*, 107838.
39. Li, H.; Li, X.; Fan, Q.; et al. Distributed DNN Inference with Fine-Grained Model Partitioning in Mobile Edge Computing Networks. *IEEE Trans. Mob. Comput.* **2024**, *23*, 9060–9074.
40. Liang, H.; Sang, Q.; Hu, C.; et al. DNN Surgery: Accelerating DNN Inference on the Edge through Layer Partitioning. *IEEE Trans. Cloud Comput.* **2023**, *11*, 3111–3125.
41. Wang, Z.; Li, F.; Zhang, Y.; et al. Low-Rate Feature Compression for Collaborative Intelligence: Reducing Redundancy in Spatial and Statistical Levels. *IEEE Trans. Multimed.* **2024**, *26*, 2756–2771.
42. Yuan, Z.; Rawlekar, S.; Garg, S.; et al. Split Computing with Scalable Feature Compression for Visual Analytics on the Edge. *IEEE Trans. Multimed.* **2024**, *26*, 10121–10133.

43. Zheng, G.; Wen, M.; Ning, Z.; et al. Computation-Aware Offloading for DNN Inference Tasks in Semantic Communication Assisted MEC Systems. *IEEE Trans. Wirel. Commun.* **2025**, *24*, 2693–2706.
44. Wang, W.; Zhang, Y.; Huang, R.; et al. Efficient Resource Management and Expansion Scheme for Collaborative Edge-Cloud Computing. *IEEE Trans. Mob. Comput.* **2024**, *23*, 2731–2747.
45. Hao, Z.; Xu, G.; Luo, Y.; et al. Multi-Agent Collaborative Inference via DNN Decoupling: Intermediate Feature Compression and Edge Learning. *IEEE Trans. Mob. Comput.* **2023**, *22*, 6041–6055.
46. Sa, C.-C.; Cheng, L.-C.; Chung, H.-H.; et al. Ensuring Bidirectional Privacy on Wireless Split Inference Systems. *IEEE Wirel. Commun.* **2024**, *31*, 134–141.
47. Zhao, S.; Liu, T.; Jin, H.; et al. SPViT: Accelerate Vision Transformer Inference on Mobile Devices via Adaptive Splitting and Offloading. *IEEE Trans. Mob. Comput.* **2025**, *24*, 9303–9318.
48. Gong, R.; Liu, X.; Li, Y.; et al. Pushing the Limit of Post-Training Quantization. *IEEE Trans. Pattern Anal. Mach. Intell.* **2025**, *47*, 5556–5570.
49. Zhu, X.; Li, J.; Liu, Y.; et al. A Survey on Model Compression for Large Language Models. *Trans. Assoc. Comput. Linguist.* **2024**, *12*, 1556–1577.
50. Cheng, H.; Zhang, M.; Shi, J.Q. A Survey on Deep Neural Network Pruning: Taxonomy, Comparison, Analysis, and Recommendations. *IEEE Trans. Pattern Anal. Mach. Intell.* **2024**, *46*, 10558–10578.
51. Liu, H.-I.; Galindo, M.; Xie, H.; et al. Lightweight Deep Learning for Resource-Constrained Environments: A Survey. *ACM Comput. Surv.* **2024**, *56*, 267.
52. Jin, H.; Bai, D.; Yao, D.; et al. Personalized Edge Intelligence via Federated Self-Knowledge Distillation. *IEEE Trans. Parallel Distrib. Syst.* **2023**, *34*, 567–580.
53. Wu, Z.; Sun, S.; Wang, Y.; et al. FedICT: Federated Multi-Task Distillation for Multi-Access Edge Computing. *IEEE Trans. Parallel Distrib. Syst.* **2024**, *35*, 1107–1121.
54. Yao, D.; Pan, W.; Dai, Y.; et al. FedGKD: Toward Heterogeneous Federated Learning via Global Knowledge Distillation. *IEEE Trans. Comput.* **2024**, *73*, 3–17.
55. Yang, X.; Yu, H.; Gao, X.; et al. Federated Continual Learning via Knowledge Fusion: A Survey. *IEEE Trans. Knowl. Data Eng.* **2024**, *36*, 3832–3850.
56. Zhang, H.; Tao, M.; Shi, Y.; et al. Federated Multi-Task Learning with Non-Stationary and Heterogeneous Data in Wireless Networks. *IEEE Trans. Wirel. Commun.* **2024**, *23*, 2653–2667.
57. Tan, A.Z.; Yu, H.; Cui, L.; et al. Towards Personalized Federated Learning. *IEEE Trans. Neural Netw. Learn. Syst.* **2023**, *34*, 9587–9603.
58. Liu, Y.; Liu, F.; Guo, B.; et al. Sparse-FCL: Sparse Federated Continual Learning for Evolving Mobile Edge Computing Environments. *IEEE Trans. Serv. Comput.* **2025**, *18*, 2403–2416.
59. Zuo, X.; Luopan, Y.; Han, R.; et al. FedViT: Federated Continual Learning of Vision Transformer at Edge. *Future Gener. Comput. Syst.* **2024**, *154*, 1–15.
60. Yu, D.; Shen, L.; Hao, H.; et al. MoESys: A Distributed and Efficient Mixture-of-Experts Training and Inference System for Internet Services. *IEEE Trans. Serv. Comput.* **2024**, *17*, 2626–2639.
61. Feng, X.; Feng, X.; Du, X.; et al. Adapter-Based Selective Knowledge Distillation for Federated Multi-Domain Meeting Summarization. *IEEE/ACM Trans. Audio Speech Lang. Process.* **2024**, *32*, 3694–3708.
62. Cai, W.; Jiang, J.; Wang, F.; et al. A Survey on Mixture of Experts in Large Language Models. *IEEE Trans. Knowl. Data Eng.* **2025**, *37*, 3896–3915.
63. Wang, L.; Chen, S.; Jiang, L.; et al. Parameter-Efficient Fine-Tuning in Large Language Models: A Survey of Methodologies. *Artif. Intell. Rev.* **2025**, *58*, 227.
64. Chen, Z.; Zhu, B.; Wang, J.; et al. Network Edge Inference for Large Language Models: Principles, Techniques, and Opportunities. *ACM Comput. Surv.* **2026**, *58*, 317. https://doi.org/10.1145/3809166.
65. Piccialli, F.; Chiaro, D.; Qi, P.; et al. Federated and Edge Learning for Large Language Models. *Inf. Fusion* **2025**, *117*, 102840.
66. Husom, E.J.; Goknil, A.; Astekin, M.; et al. Sustainable LLM Inference for Edge AI: Evaluating Quantized LLMs for Energy Efficiency, Output Accuracy, and Inference Latency. *ACM Trans. Internet Things* **2025**, *6*, 28.
67. Li, X.; Zhang, Y.; Wang, J.; et al. Cloud–Edge-End Collaborative Intelligent Service Computation Offloading: A Digital Twin-Driven Edge Coalition Approach for Industrial IoT. *IEEE Trans. Netw. Serv. Manag.* **2024**, *21*, 6318–6330.
68. Du, J.; Jiang, C.; Benslimane, A.; et al. SDN-Based Resource Allocation in Edge and Cloud Computing Systems: An Evolutionary Stackelberg Differential Game Approach. *IEEE/ACM Trans. Netw.* **2022**, *30*, 1613–1628.
69. Hossein Shokouhi, M.; Hadi, M.; Pakravan, M.R. Mobility-Aware Computation Offloading for Hierarchical Mobile Edge Computing. *IEEE Trans. Netw. Serv. Manag.* **2024**, *21*, 3372–3384.
70. Cohen, I.; Chiasserini, C.F.; Giaccone, P.; et al. Dynamic Service Provisioning in the Edge-Cloud Continuum with Bounded Resources. *IEEE/ACM Trans. Netw.* **2023**, *31*, 3096–3111.
71. Zeng, Z.; Zhao, Y.; Chen, X.; et al. Accelerating Federated Codistillation via Adaptive Computation Amount Allocation. *IEEE Trans. Mob. Comput.* **2025**, *24*, 5584–5597.

72. Lin, Z.; Bi, S.; Lin, X.; et al. Efficient Parallel Split Learning over Resource-Constrained Wireless Edge Networks. *IEEE Trans. Mob. Comput.* **2024**, *23*, 9224–9239.
73. Yuan, H.; Bi, J.; Wang, Z.; et al. Partial and Cost-Minimized Computation Offloading in Hybrid Edge and Cloud Systems. *Expert Syst. Appl.* **2024**, *250*, 123896.
74. Purice, D.; Barchi, F.; Röder, T.; et al. Edge AI Lifecycle Management. In *Advancing Edge Artificial Intelligence–System Contexts*; River Publishers: Gistrup, Denmark, 2024; pp. 43–64.
75. Hinder, F.; Artelt, A.; Hammer, B. One or Two Things We Know about Concept Drift—A Survey on Monitoring Evolving Environments. *Front. Artif. Intell.* **2024**, *7*, 1330257.
76. Zhou, Y.; You, C.; Huang, K. Communication Efficient Cooperative Edge AI via Event-Triggered Computation Offloading. *IEEE Trans. Commun.* **2026**, *74*, 3190–3205.
77. Bhuyan, B.P.; Mohapatra, S.S.; Panda, S.K. Neuro-Symbolic Artificial Intelligence: A Survey. *Neural Comput. Appl.* **2024**, *36*, 12809–12844.
78. Pan, Y.; Su, Z.; Wang, Y.; et al. Cloud-Edge Collaborative Large Model Services: Challenges and Solutions. *IEEE Netw.* **2025**, *39*, 182–191.
79. Renggli, C.; Rimanic, L.; Gürel, N.M.; et al. A Data Quality-Driven View of MLOps. *IEEE Data Eng. Bull.* **2021**, *44*, 11–23.
80. Wang, Z.; Li, X.; Wang, J.; et al. Energy-Minimized Partial Computation Offloading in Cloud-Assisted Vehicular Edge Computing Systems. In Proceedings of the 2023 IEEE International Conference on Networking, Sensing and Control (ICNSC), Marseille, France, 25–27 October 2023; pp. 1–6.
81. Younis, A.; Maheshwari, S.; Pompili, D. Energy-Latency Computation Offloading and Approximate Computing in Mobile-Edge Computing Networks. *IEEE Trans. Netw. Serv. Manag.* **2024**, *21*, 3401–3415.
82. Bi, J.; Wang, Z.; Yuan, H.; et al. Cost-Minimized Computation Offloading and User Association in Hybrid Cloud and Edge Computing. *IEEE Internet Things J.* **2024**, *11*, 16672–16683.
83. Fan, W.; Xiao, F.; Pan, Y.; et al. Latency-Aware Joint Task Offloading and Energy Control for Cooperative Mobile Edge Computing. *IEEE Trans. Serv. Comput.* **2025**, *18*, 1515–1528.
84. Yang, N.; Wen, J.; Zhang, M.; et al. Multi-objective Deep Reinforcement Learning for Mobile Edge Computing. In Proceedings of the 2023 21st International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt), Singapore, 24–27 August 2023; pp. 1–8.
85. Shao, J.; Mao, Y.; Zhang, J. Learning Task-Oriented Communication for Edge Inference: An Information Bottleneck Approach. *IEEE J. Sel. Areas Commun.* **2022**, *40*, 197–211.
86. Wang, Y.; Liu, H.; Cao, H. VOSA: Verifiable and Oblivious Secure Aggregation for Privacy-Preserving Federated Learning. *IEEE Trans. Dependable Secur. Comput.* **2023**, *20*, 3601–3616.
87. Zhu, Y.; Gong, J.; Zhang, K.; et al. Malicious-Resistant Non-Interactive Verifiable Aggregation for Federated Learning. *IEEE Trans. Dependable Secur. Comput.* **2024**, *21*, 5600–5616.
88. Zhou, Y.; Wang, R.; Liu, J.; et al. Exploring the Practicality of Differentially Private Federated Learning: A Local Iteration Tuning Approach. *IEEE Trans. Dependable Secur. Comput.* **2024**, *21*, 3280–3294.
89. Ma, J.; Zhou, Y.; Cui, L.; et al. An Optimized Sparse Response Mechanism for Differentially Private Federated Learning. *IEEE Trans. Dependable Secur. Comput.* **2024**, *21*, 2285–2295.
90. Liu, J.; Li, X.; Liu, X.; et al. Privacy-Preserving and Verifiable Outsourcing Linear Inference Computing Framework (PPVLC). *IEEE Trans. Serv. Comput.* **2023**, *16*, 4591–4604.
91. Kim, D.; Guyot, C. Optimized Privacy-Preserving CNN Inference with Fully Homomorphic Encryption. *IEEE Trans. Inf. Forensics Secur.* **2023**, *18*, 2175–2187.
92. Yamansavascilar, B.; Baktir, A.C.; Sonmez, C.; et al. DeepEdge: A Deep Reinforcement Learning Based Task Orchestrator for Edge Computing. *IEEE Trans. Netw. Sci. Eng.* **2023**, *10*, 538–552.
93. Li, K.; Wang, X.; He, Q.; et al. Computation Offloading in Resource-Constrained Multi-Access Edge Computing. *IEEE Trans. Mob. Comput.* **2024**, *23*, 10665–10677.
94. Hsieh, L.-T.; Liu, H.; Guo, Y.; et al. Deep Reinforcement Learning-Based Task Assignment for Cooperative Mobile Edge Computing. *IEEE Trans. Mob. Comput.* **2024**, *23*, 3156–3171.
95. Sun, C.; Li, X.; Wang, C.; et al. Hierarchical Deep Reinforcement Learning for Joint Service Caching and Computation Offloading in Mobile Edge-Cloud Computing. *IEEE Trans. Serv. Comput.* **2024**, *17*, 1548–1564.
96. Luo, Z.; Dai, X. Reinforcement Learning-Based Computation Offloading in Edge Computing: Principles, Methods, Challenges. *Alex. Eng. J.* **2024**, *108*, 89–107.
97. Lv, Z.; Xiao, L.; Du, Y.; et al. Efficient Communications in Multi-Agent Reinforcement Learning for Mobile Applications. *IEEE Trans. Wirel. Commun.* **2024**, *23*, 12440–12454.
98. Nguyen, D.C.; Ding, M.; Pathirana, P.N.; et al. Cooperative Task Offloading and Block Mining in Blockchain-Based Edge Computing with Multi-Agent Deep Reinforcement Learning. *IEEE Trans. Mob. Comput.* **2023**, *22*, 2021–2037.
99. Huang, T.; Fang, Z.; Tang, Q.; et al. Dual-Timescales Optimization of Task Scheduling and Resource Slicing in Satellite-Terrestrial Edge Computing Networks. *IEEE Trans. Mob. Comput.* **2024**, *23*, 14111–14126.

100. Liu, P.; An, K.; Lei, J.; et al. Computation Rate Maximization for SCMA-Aided Edge Computing in IoT Networks: A Multi-Agent Reinforcement Learning Approach. *IEEE Trans. Wirel. Commun.* **2024**, *23*, 10414–10429.
101. Gao, Z.; Yang, L.; Dai, Y. Large-Scale Computation Offloading Using a Multi-Agent Reinforcement Learning in Heterogeneous Multi-Access Edge Computing. *IEEE Trans. Mob. Comput.* **2023**, *22*, 3425–3443.
102. Li, X.; Huangfu, W.; Xu, X.; et al. Secure Offloading with Adversarial Multi-Agent Reinforcement Learning against Intelligent Eavesdroppers in UAV-Enabled Mobile Edge Computing. *IEEE Trans. Mob. Comput.* **2024**, *23*, 13914–13928.
103. Lin, Y.; Zhang, Y.; Luo, C.; et al. Dynamic Expert Pruning and Routing for Efficient On-Device Mixture-of-Experts Inference. *IEEE Internet Things J.* **2023**, *10*, 15963–15975.
104. Kong, R.; Li, Y.; Wang, W.; et al. Serving MoE Models on Resource-Constrained Edge Devices via Dynamic Expert Swapping. *IEEE Trans. Comput.* **2025**, *74*, 2799–2811.
105. Guo, B.; Zhou, C.; He, S.; et al. Mixture-of-Experts as Continual Knowledge Adapter for Mobile Vision Understanding. *IEEE Trans. Mob. Comput.* **2025**, *24*, 9868–9882.
106. Zhang, Z.; Zhao, D.; Liu, R.; et al. ACL: Adaptive Edge-Cloud Collaborative Learning for Heterogeneous Devices with Unlabeled Local Data. *IEEE Trans. Mob. Comput.* **2025**, *24*, 8152–8166.
107. Hao, Y.; Yang, S.; Li, F.; et al. Learning Adaptive Multi-Timescale Scheduling for Mobile Edge Computing. *IEEE Trans. Mob. Comput.* **2025**, *24*, 7297–7311.
108. Sharma, N.; Ghosh, A.; Misra, R.; et al. Deep Meta Q-Learning Based Multi-Task Offloading in Edge–Cloud Systems. *IEEE Trans. Mob. Comput.* **2024**, *23*, 2583–2598.
109. Wang, Z.; Wong, V.W.S. Bayesian Meta-Learning for Adaptive Traffic Prediction in Wireless Networks. *IEEE Trans. Mob. Comput.* **2024**, *23*, 6620–6633.
110. Dai, P.; Huang, Y.; Hu, K.; et al. Meta Reinforcement Learning for Multi-Task Offloading in Vehicular Edge Computing. *IEEE Trans. Mob. Comput.* **2024**, *23*, 2123–2138.
111. Liu, Z.; Huang, L.; Gao, Z.; et al. GA-DRL: Graph Neural Network-Augmented Deep Reinforcement Learning for DAG Task Scheduling Over Dynamic Vehicular Clouds. *IEEE Trans. Netw. Serv. Manag.* **2024**, *21*, 4226–4242.
112. Li, Y.; Xu, W.; Qi, Y.; et al. SR-FDIL: Synergistic Replay for Federated Domain-Incremental Learning. *IEEE Trans. Parallel Distrib. Syst.* **2024**, *35*, 1879–1890.
113. Yu, T.; Fu, K.; Wang, S.; et al. Prompting Video-Language Foundation Models with Domain-Specific Fine-Grained Heuristics for Video Question Answering. *IEEE Trans. Circuits Syst. Video Technol.* **2025**, *35*, 1615–1630.
114. Zhou, X.; Liu, M.; Yurtsever, E.; et al. Vision Language Models in Autonomous Driving: A Survey and Outlook. *IEEE Trans. Intell. Veh.* **2024**, *9*, 1–20.
115. Zhao, S.; Liu, S.; Jiang, Y.; et al. Industrial Foundation Models (IFMs) for Intelligent Manufacturing: A Systematic Review. *J. Manuf. Syst.* **2025**, *82*, 420–448.
116. AlSaad, R.; Abd-alrazaq, A.; Boughorbel, S.; et al. Multimodal Large Language Models in Health Care: Applications, Challenges, and Future Outlook. *J. Med. Internet Res.* **2024**, *26*, e59505.
117. Liu, G.-P. Digital-Twin Predictive Control of Nonlinear Systems with Time Delays, Unknown Dynamics, and Communication Delays. *IEEE Trans. Cybern.* **2024**, *54*, 7198–7210.
118. Wang, C.; Deng, Y.; Ning, Z.; et al. Building a Lightweight Trusted Execution Environment for Arm GPUs. *IEEE Trans. Dependable Secur. Comput.* **2024**, *21*, 3801–3816.
119. Chaudhari, S.; Aggarwal, P.; Murahari, V.; et al. RLHF Deciphered: A Critical Analysis of Reinforcement Learning from Human Feedback for LLMs. *ACM Comput. Surv.* **2025**, *58*, 53.
120. Cheng, Y.; Zhang, W.; Zhang, Z.; et al. Toward Federated Large Language Models: Motivations, Methods, and Future Directions. *IEEE Commun. Surv. Tutorials* **2025**, *27*, 2733–2764.
121. González Barman, K.; Lohse, S.; de Regt, H.W. Reinforcement Learning from Human Feedback in LLMs: Whose Culture, Whose Values, Whose Perspectives? *Philos. Technol.* **2025**, *38*, 35.
122. Cheng, R.; Chen, D.; Song, H.; et al. Formal Modeling and Verification Methods for the System Requirement Specifications of Train Control Systems: A Survey. *IEEE Trans. Intell. Transp. Syst.* **2025**, *26*, 1419–1440.
123. Mao, Y.; Yu, X.; Huang, K.; et al. Green Edge AI: A Contemporary Survey. *Proc. IEEE* **2024**, *112*, 880–911.
124. Ghasemi, M.; Heidari, S.; Kim, Y.G.; et al. Energy-Efficient, Delay-Constrained Edge Computing of a Network of DNNs. *IEEE Trans. Comput.* **2025**, *74*, 569–581.
125. Ma, H.; Huang, Z.; Lu, G.; et al. Greening Edge AI: Optimizing Inference Accuracy and Reducing Carbon Emissions with Renewable Energy. *IEEE Internet Things J.* **2025**, *12*, 24300–24312.