

Adaptive and Resource-efficient Agentic AI Systems for Mobile and Embedded Devices: A Survey

Sicong Liu, *Member, IEEE*, Weiye Wu, Xiangrui Xu, Teng Li, Bowen Pang, Bin Guo, *Senior Member, IEEE*, Zhiwen Yu, *Senior Member, IEEE*

Abstract—Large foundation models (FMs) such as LLMs, VLMs, diffusion models, and MLLMs have shifted AI from fragmented, task-specific models toward versatile cognitive systems. In parallel, the AI agents has been refreshed by FMs as their cognitive core, enabling autonomy, perception, planning, and self-reflection in dynamic environments. Together, these shifts open opportunities for agentic AI on mobile and edge platforms, where real-world applications demand low-latency, energy-efficient, and adaptive intelligence. However, current surveys mainly focus on static FM optimization or generic agents, overlooking mobile-specific challenges of resource constraints, runtime adaptability, and diverse conditions. This article fills that gap by providing the first systematic survey on adaptive and resource-efficient agentic AI systems on mobile/edge devices. We propose a novel taxonomy covering elastic FM inference, test-time adaptation, dynamic multimodal integration, and application-driven optimization, and we outline open issues and evaluation methodologies to inspire future research at the intersection of FMs, agents, and mobile/edge intelligence. We believe this survey can help readers to understand the connections between enabling technologies while promoting further discussions.

Index Terms—Adaptive and resource-efficient, Agentic AI, Elastic FM Inference, Test-time FM adaptation

I. INTRODUCTION

Large foundation models (FMs), including large language models (LLMs) such as GPTs and LLaMA, vision–language models (VLMs) such as CLIP and BLIP-2, diffusion models, and multimodal large models (MLLMs) such as GPT-4o and Gemini, have demonstrated remarkable general-purpose machine learning capabilities. This marks a fundamental *shift*, *i.e.*, fragmented, task-specific machine learning models (*e.g.*, CNNs [1], Transformers [2]) are converging toward versatile FMs with high-level cognition, capable of scalability, multimodal reasoning, and contextual adaptation.

Meanwhile, the concept of the AI agent was first formalized in the 1990s as autonomous systems defined by the sensing–action loop [3]. Early agents, however, built on symbolic AI or *small-scale* machine learning, lacked the reasoning power and adaptability required for long-term autonomy. With the advent of FMs as the *cognitive core*, a second *shift* has emerged, *i.e.*, AI agents can surpass traditional rule-based behaviors and achieve greater autonomy and generalization, thereby enabling perception, planning, action, and self-reflection in dynamic environments.

This dual shift is further accelerated by the growing demands of real-world mobile applications, such as autonomous driving (*e.g.*, Google Waymo [4]), robotics (*e.g.*, Meta FAIR [5]), and mobile task automation (*e.g.*, Apple

Intelligence [6]). Such applications require long-term complex operation, real-time or low-latency interaction, and robust adaptation to diverse and dynamic environments. For example, self-driving cars must process multimodal sensor streams (*e.g.*, camera, LiDAR) in vehicle systems for navigation and obstacle avoidance, while mobile task automation must respond immediately to evolving user contexts. Moreover, growing privacy concerns further highlight the importance of on-device and edge execution, where sensitive data can be processed locally without reliance on the cloud [7].

In parallel, as shown in Fig. 1, the evolution of FMs exhibits a trend reminiscent of Moore’s Law. The parameter scale required for large FM-level performance is rapidly shrinking (red lines), while the computational capacity of mobile and embedded devices continues to increase (green lines). The convergence of these trends indicates that deployable FMs on mobile and embedded platforms are becoming feasible, unlocking new opportunities. At the same time, advances in the mobile/edge ecosystem, including commercial SoCs (*e.g.*, GPUs, DSPs, NPUs) [8], edge intelligence frameworks [9], lightweight deployment libraries [10], and efficient communication protocols for multi-agent collaboration—are fostering scalable and cost-effective deployment of *agentic AI systems*.

Despite these advances, current FMs remain far from practical deployment in real-world mobile scenarios. On mobile/edge-assisted platforms, *adaptivity* and *resource efficiency* are not optional add-ons but fundamental bottlenecks. It is necessary to: *i*) restructure FMs and reasoning chains on the fly to cope with dynamic sensor data and fluctuating hardware resources; *ii*) sustain low-latency or real-time responses within stringent resource budgets; and *iii*) enable self-evolving cognitive abilities to handle non-stationary or even unseen data and tasks.

Existing surveys have addressed related aspects but remain limited. Most works [11], [12] focus on *static* FM optimization techniques (*e.g.*, distillation, quantization, pruning, fine-tuning), overlooking the need for *agentic FMs* that dynamically reconfigure structures, reasoning chains, and routing paths in response to runtime resource and environmental conditions. Although some studies explore dynamic or self-adaptive FM algorithms [13], [14], they often rely on manually designed heuristics, leaving this direction in its infancy. Meanwhile, surveys on AI agents [15] have spanned generative [16], [17], logic-based [18], [19], multimodal [15], [20], and embodied agents [21], [22], but rarely address the mobile-specific constraints of limited resources, responsiveness, and real-world adaptability.

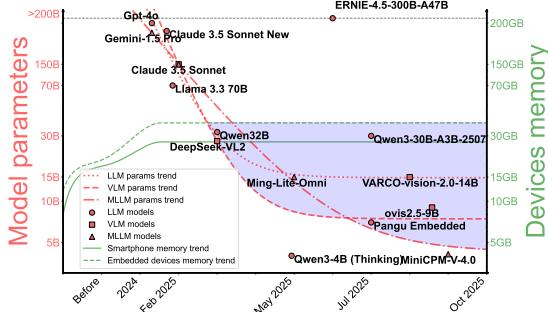


Fig. 1: Development of FMs (8-bit) and embedded hardware.

This survey aims to fill these gaps by providing a systematic study of *adaptive and resource-efficient agentic AI systems* for mobile and embedded platforms. We map enabling techniques into a unified landscape of resource efficiency and adaptivity, offering both a comprehensive taxonomy and insights into emerging challenges as follows:

- **Elastic FM Inference** (Sec. III). On mobile and embedded devices, fluctuating resource availability requires FMs to adjust their structures, often causing weight–structure *mismatches*. While retraining can mitigate these issues, it is computationally prohibitive and unsuitable for real-time applications. Also, re-compression introduces operator dependencies that complicate resource mapping, and hardware heterogeneity further hinders efficient allocation. Prior advances, including *dynamic prompts*, *selective reasoning*, *scalable depth/width*, *routing*, and *KV cache optimization*, provide initial pathways toward elastic and resource-efficient FM inference.
- **Test-time Adaptation of FM** (Sec. IV). Lightweight FMs often exhibit poor generalization, resulting in cognitive (weight) *mismatches* during long-term use. Although retraining can adapt models to data shifts and unseen tasks, resource constraints on mobile devices (*e.g.*, drones, robots) make it impractical. Moreover, the heterogeneity of distributed agent platforms reduces adaptation efficiency. We review techniques that enable online adaptation without full retraining, including *test-time prompt learning*, *parameter-efficient fine-tuning (PEFT)*, *memory augmentation*, *interactive learning*, and *system-level methods* (*e.g.*, scheduling, distributed updates), with extensions to multi-agent collaboration.

• **Dynamic Multi-modal FMs** (Sec. V). The integration of *heterogeneous* and *asynchronous* sensor streams (*e.g.*, vision, speech, LiDAR, RF) introduces both computational overhead and alignment challenges. Waiting for slow modalities inflates latency, whereas discarding them reduces accuracy, creating an inherent accuracy–latency–bandwidth trade-off. We summarize adaptive fusion strategies such as *dynamic attention*, *routing*, *alignment*, and *token compression*, which improve efficiency and scalability under mobile/edge constraints.

• **Agentic AI Applications** (Sec. VI). Real-world applications (*e.g.*, embodied agents, GUI assistants, generative agents, and personal assistive services) demand long-term operation, responsiveness, and interactive adaptability. Static FM deployments struggle to satisfy these requirements due to rigid pipelines and limited resource awareness. We demonstrate how *elastic inference* and *adaptive retraining* can integrate FM capabilities into context-sensitive, resource-aware applications,

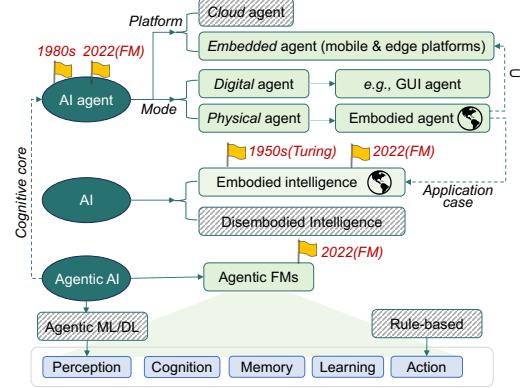


Fig. 2: Related concepts.

emphasizing the role of *application-driven optimization* in bridging enabling techniques with practical deployment.

In summary, our contributions are threefold:

- To the best of our knowledge, this is the first systematic survey of adaptive and resource-efficient agentic AI systems on mobile & embedded devices, outlining pathways for re-mapping FM structures, cognition, and hardware resources within a hardware–software spectrum.
- We propose a novel taxonomy of enabling techniques, spanning elastic inference, test-time adaptation, and dynamic multimodal integration, clarifying trade-offs in accuracy, latency, communication, and energy efficiency.
- We identify open issues in adaptive and resource-efficient agentic AI systems and outline potential research directions to guide innovation in FM architecture, algorithm–system co-design, and mobile/edge deployment.

The rest of this paper is organized as follows. Sec. II reviews the fundamentals; Sec. III and Sec. IV survey elastic inference and test-time adaptation; Sec. V discusses dynamic multimodal FMs; Sec. VI highlights representative applications. Open challenges and evaluation methodologies are presented in Sec. VII, and conclusions are drawn in Sec. VIII.

II. FUNDAMENTALS AND OVERVIEW

This section provides an overview of adaptive and resource-efficient agentic AI systems on mobile and embedded devices, and clarifies their relation to existing concepts.

A. Related Concepts

Agentic AI systems build upon, and intersect with, several related notions (Fig. 2).

Mobile and embedded devices are ubiquitous hardware platforms (*e.g.*, CPUs, GPUs, DSPs, NPUs, MCUs) that operate under strict compute, memory, and energy constraints (*e.g.*, smartphones, UAVs, in-vehicle units). They provide the execution platform.

AI agents, formalized in the 1990s [3], follow the *sensing–decision–action loop* and may exist in *digital* (*e.g.*, GUI assistants) or *physical* (*i.e.*, embodied agents) forms, and can be deployed across *cloud*, *edge*, and *mobile* environments. This survey focuses on *edge and mobile settings*, where resource efficiency is a critical bottleneck.

Embedded AI agents run on resource-constrained mobile/edge platforms, requiring *resource-efficient* techniques for sustained operation. *Embodied AI agents* are equipped with sensors and actuators for direct physical environmental interaction (e.g., drones, autonomous vehicles). Conceptually, embodied agents are a subset of embedded agents (*i.e.*, embodied \subset embedded).

Embodied intelligence, envisioned by Turing in the 1950s [23], emphasizes cognition through physical perception and interaction, today instantiated by embodied agents integrated with FMs.

Agentic AI represents a paradigm shift since 2022 because FMs (e.g., LLMs, VLMs, MLLMs) extend agents beyond *perception* to *high-level cognition*, including reasoning, task decomposition, and decision-making. When augmented with planning, memory, tool use, and reflection, FMs function as the *cognitive core of AI agents*, enabling autonomy and generalization. An *agentic AI system* typically comprises one or multiple FM-powered agents, coordinated through scheduling and communication, to interact with heterogeneous systems, adapt to dynamic environments, and collaborate with humans. This constitutes the scope of this survey.

B. Our Scope

This survey focuses on *adaptive, resource-efficient agentic AI systems* for *mobile and embedded platforms*, conceptualized across five interrelated levels (Fig. 3). We emphasize mobile and embedded settings because the tension between large-scale FMs and constrained resources is the primary bottleneck, further compounded by rising privacy concerns that motivate on-device or edge execution rather than cloud reliance.

- *Hardware level*: resource-constrained mobile/embedded devices or edge servers serve as the physical platform.
- *System scheduling level*: FMs are compiled into DAGs and dynamically mapped across heterogeneous compute/memory resources via *frontend-backend co-compilation*, maximizing hardware utilization.
- *Model level*: FMs enhanced with planning, memory, and reasoning (*agentic FMs*) act as the cognitive “brain” for perception and decision-making.
- *Agent level*: AI agents emerge from the integration of platforms, FMs, and sensing-action loops. On embedded devices, they form *embedded agents*; equipped with physical sensors and actuators, they become *embodied agents*.
- *Network level*: agents coordinate through scheduling and communication to form distributed *agentic AI systems*.

In this survey, we exclude resource-rich cloud deployments, as they face fewer efficiency constraints. Nevertheless, the techniques discussed here remain broadly relevant across the cloud-edge-mobile continuum.

C. Why Adaptivity and Resource-Efficiency Matter

Unlike resource-rich clouds, these systems must remain responsive under real-world *diversity* and *dynamics*. Challenges arise from four-fold: *i*) fluctuating hardware resources due to contention, throttling, and heterogeneity; *ii*) unstable mobile

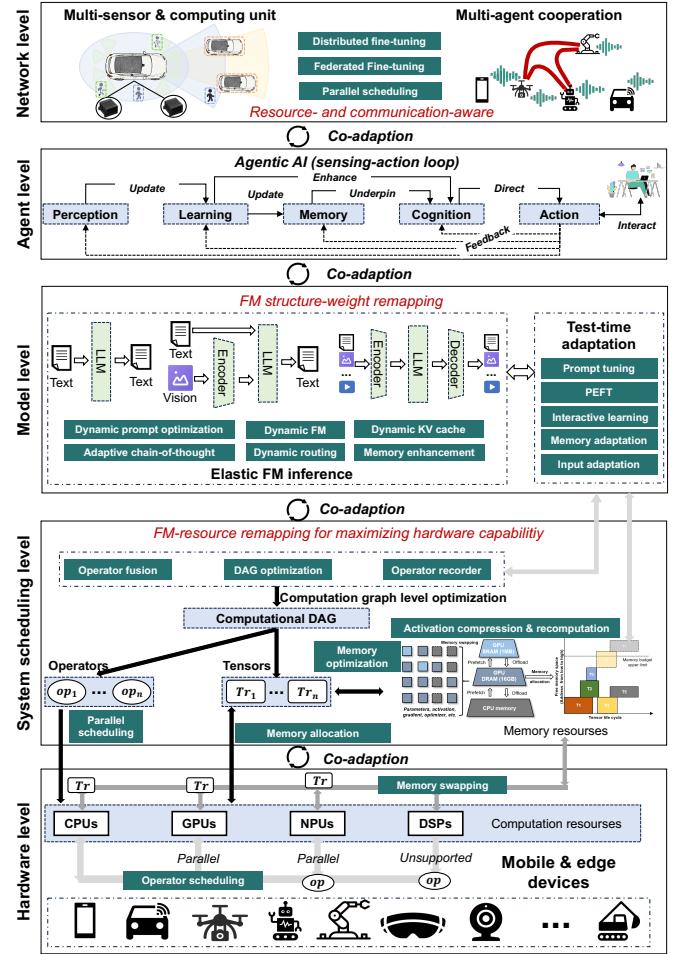


Fig. 3: Cross-level landscape of agentic AI systems.

networks with variable latency and bandwidth; *iii*) dynamic inputs and tasks (e.g., multimodal bursts, environmental shifts, heterogeneous user demands) requiring flexible FM structures and operator flows; and *iv*) long-running, open-world operations where agents collaborate and process distributed multimodal streams.

In such contexts, *static* FM structure–weight–resource mappings degrade quickly. Full retraining is both costly and too slow, while structural adjustments introduce new operator dependencies and resource mismatches exacerbated by hardware heterogeneity. What is needed are *elastic reconfiguration* mechanisms to realign FM structures with resources, and *efficient adaptation* techniques to update parameters under nonstationary tasks and data.

D. Characteristics of FM-based Agentic AI Systems

By leveraging FM backbones, agentic AI systems inherit distinctive traits that enhance *adaptivity* and *resource efficiency*, surpassing traditional DL-based mobile systems [24], [25] that are constrained to narrow, pre-defined tasks. FMs, pre-trained on large-scale multimodal data and unified transformer architectures (e.g., LLaMA-2 [26], ViT-G [2]), exhibit emergent reasoning even at moderate scales [27], [28]. These properties empower agents to generalize via prompting [29], decompose complex goals [30], incorporate contextual mem-

ory [31], and refine behavior through human or environmental feedback [32], [33]. Concretely, FM-powered agentic AI systems exhibit four key capabilities that support *complex* tasks:

- *Cross-task generalization*: Pre-trained FMs (*e.g.*, LLMs, VLMs, MLLMs) handle diverse modalities with minimal retraining cost, offering robust adaptability.
- *Goal decomposition*: High-level instructions (*e.g.*, *Plan a trip*) are decomposed into sub-goals (*e.g.*, booking, transportation) via chain-of-thought reasoning.
- *Contextual memory*: Lightweight episodic memory structures [34] capture and reuse preferences and past interactions, enabling personalization and continuity.
- *Feedback integration*: Reinforcement learning from human feedback (RLHF) combined with environmental signals (*e.g.*, battery, network state) refines behavior online, balancing accuracy, latency, and energy efficiency.

These underscore adaptivity and efficiency for complex tasks, but also pose challenges. FMs are resource-intensive, planning needs elastic inference, memory entails efficiency–accuracy trade-offs, and feedback requires online adaptation.

E. Taxonomy of Enabling Techniques

We summarize four categories of enabling techniques for adaptive and resource-efficient agentic AI systems:

- *Elastic FM Inference* (Sec. III). Elastic inference dynamically adapts FM architectures, reasoning *depth*, *computation*, and *communication* at runtime to meet accuracy–latency–energy goals under fluctuating device and network constraints. Unlike static pipelines, it treats execution as continuously tunable across memory, compute, and bandwidth. Techniques include *dynamic prompts* [35], [36] to cut input redundancy, *adaptive CoT* [37]–[39] for selective reasoning, *scalable architectures* [40], [41], *dynamic routing* [13], [42], [43], and *KV cache management* [44], [45] for long-horizon memory.
- *Test-time Adaptation of FM* (Sec. IV). To remain robust under data shifts, *partial observability*, and *long-horizon tasks*, agentic AI systems must adapt FMs’ parameter weights *online* without full retraining. This can be achieved via *algorithmic strategies*—prompt learning [46]–[48], PEFT [49], [50], memory augmentation [51]–[53], interactive learning [33], [54], [55], and *system-level techniques* such as memory management [56]–[58], scheduling [59]–[61], and distributed adaptation [62], [63]. Beyond single agents, collaborative test-time adaptation leverages data, pipeline, and heterogeneous processor parallelism in multi-agent systems.
- *Dynamic Multi-modal FMs* (Sec. V-A). Mobile and edge agents must integrate heterogeneous sensor streams (*e.g.*, vision, speech, LiDAR, RF), which increases computation, memory, and alignment overhead. Dynamic multi-modal FMs employ *dynamic attention* [64], [65], *routing* [66], [67], *alignment* [68], [69], and *token compression* [70] to improve efficiency, consistency, and scalability under tight constraints.
- *Agentic AI Applications* (Sec. VI). Applications such as embodied agents [71], [72], GUI agents [73], [74], gener-

ative agents [75], [76], and personal assistive agents [77], [78] demonstrate how elastic inference and adaptive re-training integrate FM capabilities into resource-aware, context-sensitive services, highlighting the importance of *application-driven optimization*.

Workflow. An FM-powered agentic AI system follows a closed-loop workflow of *perception*, *cognition*, *memory*, *learning*, and *action* across one or multiple devices (Fig. 4). Each module is optimized as follows:

- *Perception*: ingests multi-modal signals (*e.g.*, vision, speech, sensors, text) and encodes them into embeddings. Optimizations such as token quantization, prompt pruning, and lightweight feature extraction reduce latency and energy on constrained mobile/edge devices (see Sec. III-A2, Sec. III-C2, Sec. III-C3).
- *Cognition*: performs reasoning, goal decomposition, and decision planning via transformer layers, CoT prompting, and hierarchical planners. Dynamic routing, adaptive attention, and elastic CoT enable scalable inference under varying resource and performance demands (Sec. III-D, Sec. III-C1, Sec. III-B).
- *Memory*: maintains episodic and semantic context through key–value attention, retrieval, and differentiable updates. Efficiency is improved via quantization, knowledge distillation, and KV cache management (Sec. III-C3, Sec. III-C4, Sec. III-E, Sec. IV-C).
- *Agent Learning*: adapts to non-stationary data/tasks through test-time adaptation, policy optimization, and RLHF. Interactive learning, prompt tuning, and PEFT also provide efficient adaptation under tight budgets (Sec. IV-A, Sec. IV-B, Sec. IV-D).
- *Action*: executes decisions and generates outputs via policy networks, decoders, and diffusion models. Early exiting and layer skipping enhance responsiveness on mobile/edge hardware (see Sec. V-A2).
- *System scheduling* orchestrates elastic inference and adaptation through operator fusion, runtime scheduling, heterogeneous resource allocation for inference (see Sec. III-F1, Sec. III-F2, Sec. III-F3) and memory scheduling, computation graph optimization for retraining (Sec. IV-E1, Sec. IV-E2).

Together, they push the trade-off boundary between accuracy, latency, memory, and energy, enabling practical deployment of agentic AI systems on mobiles, wearables, and edge.

F. Performance Metrics

In both inference and retraining, agentic AI systems must balance user goals (*e.g.*, accuracy, latency, energy efficiency) with dynamic device constraints (*e.g.*, memory hierarchy, battery life). We summarize the key metrics below.

Accuracy. Accuracy is fundamental for reliable task execution. Beyond standard measures (*e.g.*, classification accuracy, perplexity, BLEU/ROUGE, factuality [29]), task-level indicators such as planning correctness, termination error rate, and task success rate can evaluate the end-to-end reliability.

Latency. Low latency is essential for interactive responsiveness. Besides *end-to-end latency*, fine-grained metrics include:

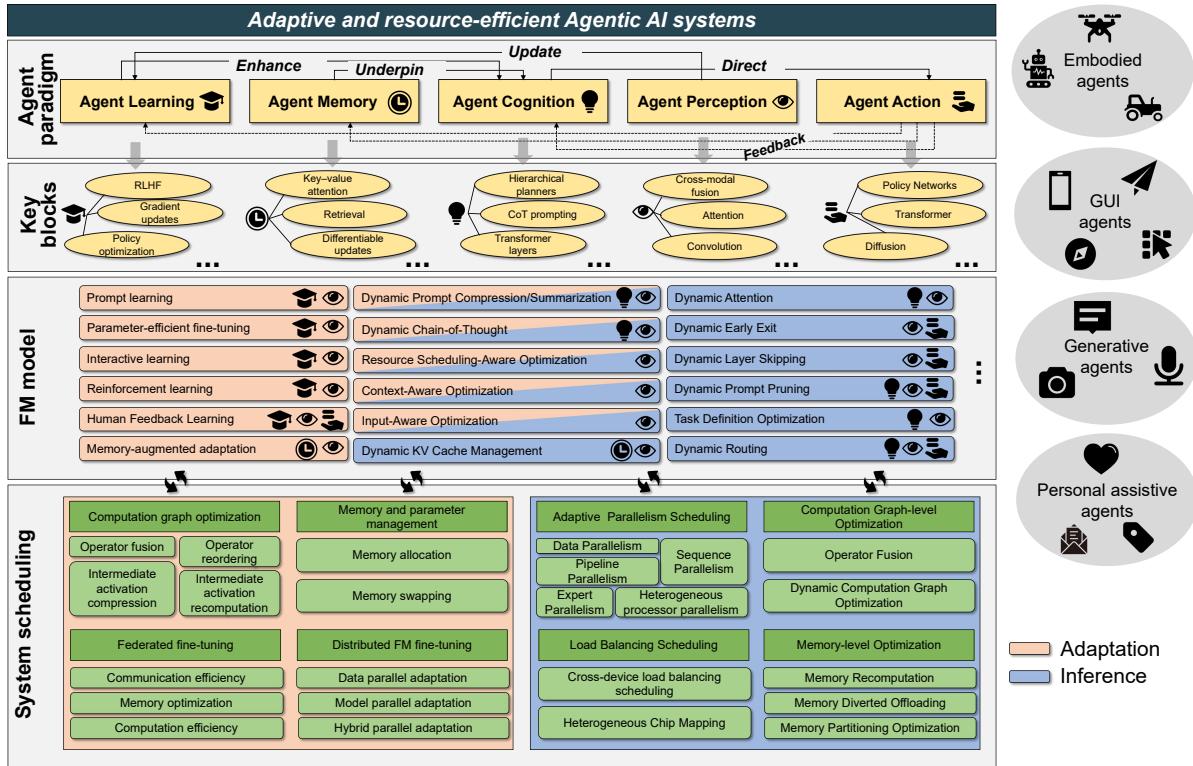


Fig. 4: Dynamically adaptive and resource-efficient agentic AI system workflow.

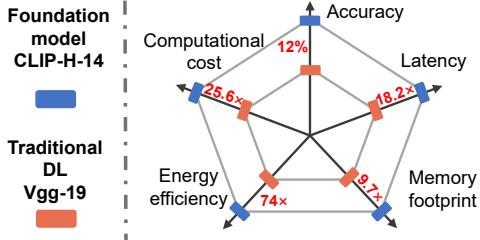


Fig. 5: Performance comparison of FMs and traditional DL.

Time To First Token (TTFT) [79], [80], reflecting initial responsiveness; *Time Per Output Token (TPOT)* [80], indicating sustained throughput; and *Time Between Tokens (TBT)* [80], where long gaps reduce naturalness.

Memory footprint. Large FM parameters and activations make memory occupancy a primary bottleneck. Key metrics include total memory budget (executability), SRAM utilization (reducing DRAM traffic), and cache hit rate (data reuse efficiency) [59], [81], [82]. Efficient memory access is also critical for both latency and energy [83].

Computational Cost. Measured by multiply-accumulate (MAC) operations or FLOPs, computational cost directly impacts latency and energy. Retraining is significantly more expensive than inference due to additional backward passes.

Energy Efficiency. Measured by multiply-accumulate (MAC) operations or FLOPs, computational cost directly impacts latency and energy. Retraining is significantly more expensive than inference due to additional backward passes.

As shown in Fig. 5, VGG-19 [84] trained on ImageNet-1K achieves 72.37% Top-1 accuracy, while CLIP (ViT-H-14-378) [85] attains 84.37% zero-shot accuracy without su-

pervised training on ImageNet. This accuracy advantage of foundation models is offset by substantial overheads: CLIP requires 18.2 \times FLOPs (GMacs), 9.7 \times latency, 74 \times peak memory, and 25.6 \times energy compared to VGG-19, intensifying the deployment challenges on constrained platforms. The system must therefore perceive both environmental inputs and resource availability, and leverage timely online performance prediction and validation to balance accuracy, latency, memory, and energy cost for practical deployment.

III. ELASTIC FM INFERENCE IN AGENTIC AI SYSTEMS

It is critical for agentic AI systems to meet diverse user demands (*e.g.*, accuracy, latency, energy efficiency) and adapt to dynamic device/network constraints (*e.g.*, memory, battery, bandwidth). Unlike static pipelines, elastic FM inference derives a new paradigm where *FM structure*, *reasoning depth*, *computation cost*, and *resource allocation* become *runtime-adaptive* and *tunable*, enabling sustainable deployment on mobile, wearable, and distributed agent platforms. As shown in Fig. 6, we summarize recent advances to highlight multiple aspects of elasticity: *i*) *dynamic prompt optimization* reduces input complexity while preserving accuracy; *ii*) *adaptive chain-of-thought* expands reasoning selectively; *iii*) *dynamic FM models* allow scalable depth/width; *iv*) *dynamic routing* enables path selection at token or layer level; and *v*) *dynamic KV cache management* controls memory for long-horizon interactions. They define a novel landscape that pushes beyond static inference, reshaping FMs into adaptive and resource-efficient backbones for agentic AI systems.

TABLE I: Summary of dynamic prompt optimization techniques for efficient inference in agentic AI systems.

Categories		Technique highlight	Year	Ref
Dynamic prompt optimization (\$III-A)	Dynamic prompt compression (\$III-A1)	Compresses documents into concise textual summaries. Compress text into a concise summary vector.	2024 2023	[36] [86]
	Dynamic prompt pruning (\$III-A2)	Compresses long prompts into a small set of “gist” tokens via modified attention masks. Train a meta-controller to predict the number of context examples required.	2023	[51]
	Adaptive retrieval-augmented generation (\$III-A3)	Uses reinforcement learning to directly edit discrete prompts for efficient compression . Converts GUI elements into a lightweight HTML tag system with functional attributes.	2024 2024	[87] [88]
	Task definition optimization (\$III-A4)	Combine parametric memory and non-parametric memory . Introduces a self-reflection mechanism with dynamic reflection tokens for real-time accuracy. Employs a prospective prediction mechanism to activate knowledge retrieval based on upcoming text analysis. Use the prediction results of LLM to supervise the retrieval module. Leverages language models’ self-reflection by converting feedback into verbal guidance. Synergizes reasoning and acting by interleaving thought and action generation.	2020 2024 2023 2023 2023 2023	[89] [90] [91] [92] [93] [94]

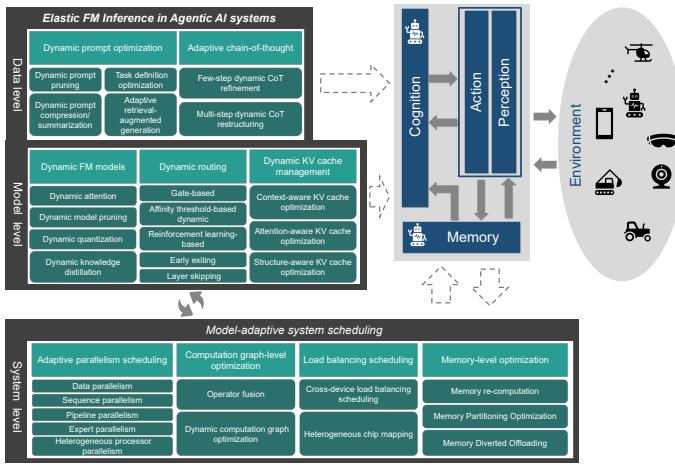


Fig. 6: Overview of elastic FM inference techniques.

A. Dynamic Prompt Optimization

Due to the fundamental differences in input processing paradigms between FMs and traditional DL models, *dynamic prompt optimization* for FMs has positioned as an emerging research focus. Prompting is the primary way to guide responses effectively, but it consumes precious space in the input context window, and repeatedly encoding the same prompt leads to inefficiencies in computation [51]. Thus adaptive FM prompt optimization represents a novel paradigm addressing three core mobile challenges, *i.e.*, real-time application demands, environmental variability, and flexible user interactions. This approach enhances Agentic AI’s expressive power, enabling real-time adjustment of guidance strategies. We systematically categorize dynamic prompting strategies for Agentic AI systems into four, *i.e.*, *dynamic prompt compression* [36], [51], [86], *prompt pruning* [35], [87], [88], *adaptive Retrieval-Augmented Generation (RAG)* [89]–[92], and *task definition Optimization* [51], [93], [94](in Tab. I).

1) *Dynamic Prompt Compression/Summarization*: Dynamic prompt compression condenses long prompts into compact summaries that retain semantic relevance, serving as learnable soft prompts or direct FM inputs [36], [51], [86]. Approaches include *summarization accumulation*, *segmentation*, *task-adaptive extractive/abstractive compressors*, and *attention-mask modification* (Fig. 7a). AutoCompressors [86] compresses text into summary vectors used as soft prompts, trained via unsupervised objectives and optimized with summarization accumulation [95] and random segmentation. RE-COMP [36] learns task-aware extractive and abstractive compressors for retrieval-augmented LMs, balancing compression

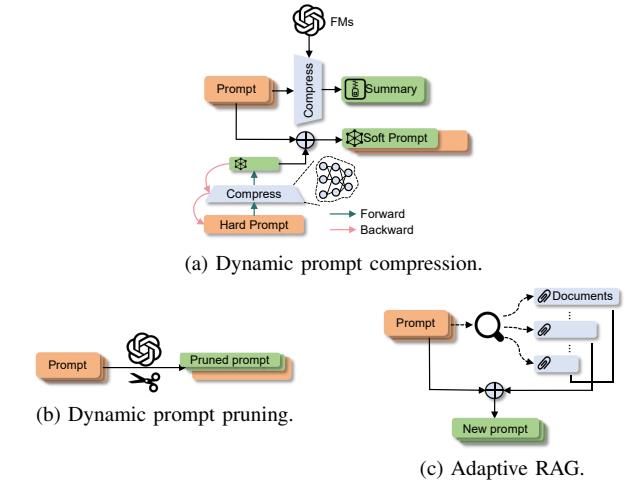


Fig. 7: Illustration of dynamic prompt optimization.

rates with computation through selective document enhancement. Gisting [51] achieves up to 26× compression by altering attention masks during fine-tuning, condensing prompts into gist tokens without retraining.

2) *Dynamic Prompt Pruning*: Dynamic prompt pruning [35], [87], [88] selectively removes unimportant prompt content to reduce complexity, complementing compression and summarization methods (Fig. 7b). Strategies include meta-controller-based context allocation [35], token-level pruning [87], and UI element pruning [88]. DYNACL [35] trains a FLAN-T5 meta-controller to predict context size per input, balancing accuracy with compute budget. PCRL [87] applies reinforcement learning to edit prompts directly, pruning redundant tokens without model gradients or labeled data. AutoDroid [88] prunes GUI/HTML elements by merging functionally equivalent nodes and discarding non-informative containers, lowering LLM processing overhead in mobile interaction tasks. Collectively, these methods adapt prompt length dynamically, improving efficiency while preserving task relevance.

3) *Adaptive Retrieval-Augmented Generation (RAG)*: Retrieval-Augmented Generation (RAG) enhances FM accuracy and consistency by integrating external knowledge bases on demand [89], reducing long-context inputs while ensuring relevant retrieval at the right time (Fig. 7c). Its advantages lie in *indexing knowledge for efficient reasoning* and *on-demand retrieval* for adaptive context construction. Recent advances introduce *adaptive* mechanisms to refine retrieval and generation. Self-RAG [90], [96] adds *self-reflection tokens* for real-time quality control. FLARE [91], [97] employs

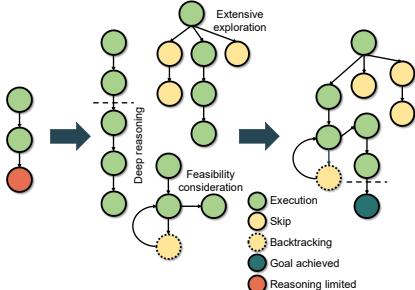


Fig. 8: Compared to short CoT with few-step reasoning, long CoT features three characteristics: 1. deep reasoning, 2. extensive exploration, 3. feasibility consideration.

prospective prediction, triggering retrieval based on upcoming semantic needs. REPLUG [92], [98] applies a *black-box strategy*, enhancing frozen FMs with tunable retrieval modules. AutoDroid [88] integrates commonsense and app-specific knowledge via *dynamic analysis*, enabling task automation across arbitrary Android apps.

4) *Task Definition Optimization*: Task definition optimization seeks to refine prompts by *compressing* or *restructuring* them to improve efficiency without sacrificing effectiveness [51], [93], [94]. In particular, Gisting [51] modifies *attention masks* during instruction fine-tuning to encode prompts into compact "Gist Tokens," which can be cached and reused then, saving computation and context window space. Reflexion [93] *distills* failed trajectories into concise natural language summaries, integrating lessons learned into subsequent prompts via semantic compression. However, overly aggressive compression may induce hallucinations or loss of task control. To address this, ReAct [94] interleaves *think–act–observe* trajectories in prompts, guiding explicit reasoning and interaction steps. This synergy between reasoning and acting enables dynamic planning, fact acquisition, and strategy adjustment, reducing factual hallucinations in traditional chain-of-thought prompting.

Discussion. Actuallu, dynamic prompting in agentic AI follows two paradigms (Fig. 7), *i.e.*, *rule-based* and *data-driven*. The *rule-based* paradigm uses deterministic templates and adaptive triggers, ensuring verifiable behavior and ms-level latency for time-sensitive tasks, but struggles with unseen or dynamic scenarios. In contrast, the *data-driven* paradigm leverages trainable models and retrieval augmentation for adaptive prompt optimization in knowledge-rich settings, offering flexibility at the cost of higher complexity and weaker determinism. These complementary strengths provide a framework for aligning prompting strategies with diverse agentic AI application requirements.

B. Adaptive Chain-of-Thought

Chain of Thought (CoT) enables FMs to *reason incrementally*, decomposing complex questions into stepwise rationales rather than producing direct answers. Dynamic CoT extends this capability with *adaptive control*, allowing models to adjust reasoning depth and structure in response to task complexity, resource budgets, and stability demands. As summarized in

Tab. II, approaches fall into two categories: *few-step adaptation*, which dynamically selects shorter or longer reasoning chains at runtime, and *multi-step restructuring*, which revises or branches reasoning trajectories during execution for more reliable outcomes.

1) *Few-Step Dynamic CoT Refinement*: Few-Step Dynamic CoT refers to a *shallow, linear* reasoning process composed of a limited number of sequential nodes. Each step unidirectionally leads to the next without repetition or backtracking, making it suitable for simple, well-defined problems that require high speed and low resource consumption. It provides fast, resource-efficient reasoning, and its refinements focus on improving adaptability, efficiency, and structural robustness for simple or time-critical agentic AI tasks. Specifically, recent studies explore three refinement directions, *i.e.*, *task-driven adaptation* [37], [99], [107], *prompt-based efficiency* [38], [100], and *structural optimization* [39], [101].

a. *Task-driven adaptation*. Several methods enhance LLMs' reasoning across diverse tasks. AgentInstruct [37] guides zero-shot reasoning via agent-generated task-specific instructions. GeM-CoT [99] selects demonstrations by question type for stronger generalization. Automatic CoT [107] retrieves similar examples or falls back to zero-shot reasoning to update its demo pool. COSP samples multiple reasoning paths and re-prompts with the best candidate. Reprompting iteratively learns effective prompts via Gibbs sampling.

b. *Prompt-based efficiency*. Skeleton-of-Thought [38] decomposes answers into frameworks first, enabling parallel generation and reducing latency. Fig. 9 illustrates the working principle of SoT. Adaptive-Consistency [100] improves self-consistency sampling by dynamically adjusting sample size with lightweight stopping criteria.

c. *Structural optimization*. Scratchpad [39] introduces an intermediate buffer for storing reasoning steps, allowing adaptive allocation of computation. Self-Notes [101] extends this idea by letting LLMs generate notes in real time during reading and answering, enhancing memory capacity and multi-step reasoning ability.

2) *Multi-Step Dynamic CoT Restructuring*: Unlike the fixed or shallow reasoning of short CoT, multi-step CoT enhances complex task solving by introducing *dynamic adaptability* through three mechanisms, *i.e.*, *depth expansion* [108], [109], *breadth exploration* [103], [104], and *self-refinement* [93], [105]. Fig. 8 illustrates these typical characteristics of multi-step CoT. This paradigm is particularly effective in *mathematical*, *programming*, and *cross-domain* reasoning where long, adaptive chains are required.

a. *Depth expansion*. Multi-step CoT can extend reasoning depth beyond the node limits of short chains, developing *incremental hierarchical logic* only when problem complexity demands it. For example, Program-Aided Language Models (PAL) [102] dynamically invoke program execution to extend reasoning, and MathPrompter [109] adaptively expands reasoning depth for difficult mathematical problems.

b. *Breadth exploration*. Moving from linear to tree-structured reasoning ($n_i \rightarrow n_{i+j}$), multi-step CoT can perform *dynamic branching* to generate and evaluate parallel reasoning paths, pruning or prioritizing them based on resource budgets.

TABLE II: Summary of adaptive Chain-of-Thought (CoT) techniques for efficient inference in agentic AI systems.

Categories		Technique highlight	Year	Ref
Adaptive chain-of-thought (§III-B)	Few-step dynamic CoT refinement (§III-B1)	Construct an autonomous agent to guide LLM reasoning.	2024	[37]
		Obtain or construct corresponding demonstrations.	2024	[99]
		First output the answer skeleton and then generate each key point.	2024	[38]
		Dynamically adjust the number of samplings for each question.	2023	[100]
		Write the intermediate steps into the scratchpad.	2021	[39]
	Multi-step dynamic CoT restructuring (§III-B2)	Think and record thoughts during the process of reading the context.	2023	[101]
		Synergizes large language models with program interpreters .	2023	[102]
		Explore and evaluate multiple reasoning paths.	2023	[103]
		Represents “thoughts” as a directed graph with dependencies and feedback.	2024	[104]
		Generates and utilizes self-feedback without additional training.	2023	[105]
		Combines reinforcement learning-driven rule rewards and tree search for deep logical exploration.	2025	[106]

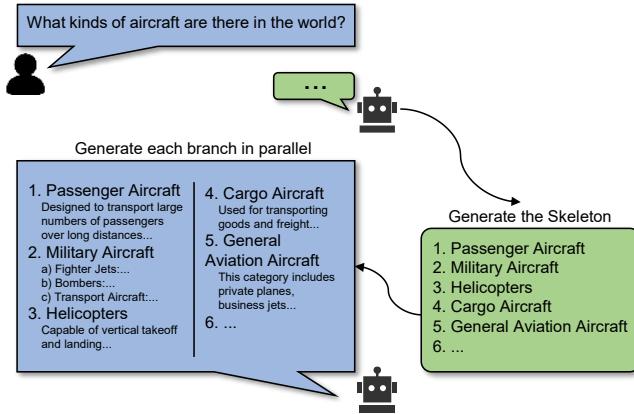


Fig. 9: SoT generates an answer framework and then generates finer branches in parallel to achieve acceleration.

Examples include Tree-of-Thoughts (ToT) [103] and Graph-of-Thoughts [104], which adaptively expand or collapse search paths to balance coverage and efficiency.

c. Self-refinement. Feedback loops enable dynamic verification and correction of intermediate steps, reallocating computation when inconsistencies arise. Approaches such as Reflexion [93] adapt reasoning by leveraging past errors, while Self-Refine [105] iteratively critiques and edits outputs until convergence, embodying dynamic correction.

These mechanisms can also be combined adaptively. For example, DeepSeek-R1 [106] integrates reinforcement learning with multi-stage reward signals (*depth expansion*), tree search-based exploration (*breadth exploration*), and dynamic feedback with self-correction (*self-refinement*), achieving deeper reasoning hierarchies and higher-precision outputs in challenging mathematical and programming tasks.

C. Dynamic FM Model

Dynamically scaling FM models is a basic solution to resource limitations in embedded and embodied agents. Agent platforms always face strict power budgets and latency-sensitive demands, making structural optimization of FMs essential. We summarize main directions of dynamic model design: (i) *dynamic attention* [40], [110]–[112] allocates elastic attention computation for multi-granularity feature perception; (ii) *dynamic pruning* [41] removes redundant structures at runtime to cut FLOPs; (iii) *dynamic quantization* [58], [113], [114] adapts precision levels to balance efficiency and accuracy; and (iv) *dynamic knowledge distillation* [115], [116] transfers knowledge into compact models for efficient

inference(in Tab. III). Beyond these structural approaches, mechanisms include (v) *dynamic routing* for adaptive path selection and resource scheduling, and (vi) *dynamic KV cache management* for compressing long-sequence caches, jointly improving efficiency under tight resource budgets.

1) Dynamic Attention: Multi-head attention (MHA) is a core module in both *decoder-only* and *encoder-decoder* FMs, capturing complex sequence dependencies. Dynamic MHA has been enhanced through *sparse attention* [59], [110], [117] and *hierarchical attention* [118]–[120], which reduce redundant operations and improve adaptability. *Sparse attention* limits the number of active connections in the attention matrix, while *hierarchical attention* prioritizes critical layers or regions, jointly improving memory and computational efficiency.

a. Sparse attention. Sparse attention *selectively activates* critical units, reducing complexity while preserving representational power (Fig. 10). Key strategies include differentiable routing (*e.g.*, gated networks) for *dynamic unit selection* and *input-adaptive allocation*. FLASH [40] integrates gating with simplified attention via the Gated Attention Unit (GAU), replacing costly multi-head softmax with a lightweight single-head design. SPARSEK Attention [110] combines a scoring network with differentiable top-k masking, selecting a fixed number of KV pairs per query to achieve linear time and constant memory. Squeezed Attention [117] applies query-aware dynamic KV selection with Softmax-based sparsity control, outperforming fixed pruning (*e.g.*, SnapKV) by better balancing accuracy and efficiency.

b. Focused dynamic attention. It is another common approach that partitions sequences into *overlapping* or *non-overlapping windows* with *predefined* or *dynamic* sizes. Each query attends to local neighbors and a few global nodes, while cross-window correlations are built via relative positional encoding and window shifting. Routing Transformer [111] combines content-based clustering with local and temporal sparse attention, improving flexibility and efficiency. NSA [112] employs fixed windows with attention aggregation, using coarse-grained tokens for global context, fine-grained tokens for details, and sliding windows for local dependencies. These designs preserve global context while overcoming the limitations of static sparse patterns in long-sequence modeling.

c. Hierarchical/Progressive attention. Hierarchical attention organizes computation into *layered structures* to capture multi-scale features. Unlike sparse attention, it stresses inter-layer collaboration, lower layers extract fine-grained details, while higher layers encode coarse-grained semantics. Two main strategies are used, *i.e.*, *decoupled processing* and *progressive granularity*. *First, decoupled attention* separates

TABLE III: Summary of dynamic FM model structures for elastic inference in agentic AI systems.

Categories		Technique highlight	Year	Ref
Dynamic FM model (\$\S\$III-C)	Dynamic attention (\$\S\$III-C1)	By unifying gating mechanisms through the Gated Attention Unit(GAU).	2022	[40]
		Integrate a scoring network and a top-k mask to select KV pairs for each query.	2024	[110]
		By controlling sparsity through a threshold.	2024	[117]
		Content-based sparse attention, incorporate a sparse routing module.	2021	[111]
		Achieve sparsity through a fixed window combined with attention aggregation.	2025	[112]
		Decouple storage demands across attention heads based on their operational roles.	2024	[118]
	Dynamic model pruning (\$\S\$III-C2)	Make coarse-grained and fine-grained attention divisions for tokens.	2023	[119]
		Use the Gaussian probability density function to compute attention weights.	2024	[120]
	Dynamic quantization (\$\S\$III-C3)	Enables one-shot pruning of large GPT models to 50% sparsity.	2023	[41]
		Employs a lightweight prediction module to dynamically prune redundant tokens.	2021	[121]
		Leverages the early-bird lottery ticket hypothesis to identify sub-networks.	2021	[122]
	Dynamic knowledge distillation (\$\S\$III-C4)	Protects salient weights based on activation distribution.	2024	[113]
		Data-free quantization-aware training supports 8-bit weight and activation quantization.	2023	[123]
		Uses tensor decomposition and mixed precision for low-bit inference on CPU/NPU.	2025	[114]

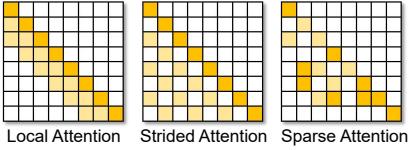


Fig. 10: Sparse attention.

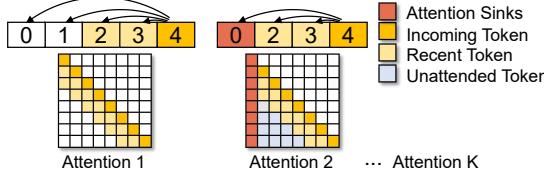


Fig. 11: Hierarchical attention.

functions or context spans for efficiency. DuoAttention [118] assigns retrieval heads to long-range KV caches and streaming heads to recent tokens, showing that only a few retrieval heads suffice for long contexts, enabling pruning and selective memory usage. *Second, progressive attention* builds multi-granularity layers. Early stages apply fine-grained attention (*e.g.*, token/pixel), while later stages use coarse-grained forms (*e.g.*, sentence/region). BiFormer [119] partitions tokens by granularity for efficient routing, while GAAM [120] employs Gaussian-based heads that dynamically adjust focus via learned means and variances, providing distribution-aware expressiveness.

2) Dynamic Model Pruning: Model pruning reduces resource cost and latency by removing redundant parameters or structures. The key idea is to adaptively eliminate weights or neurons with minimal contribution to specific inputs, tasks, or outputs. It typically include two types: *Unstructured pruning* offers fine-grained control by removing individual weights, but the resulting sparse matrices often underutilize hardware [41]. *Structured pruning*, in contrast, removes whole neurons [122], attention heads [126], or channels [121], [127], preserving dense formats that map more efficiently to hardware.

3) Dynamic Quantization: Quantization reduces overhead by lowering parameter precision. Converting FP32 weights to INT8, for example, cuts memory by 4× and accelerates inference with integer operations. Two main approaches exist: *post-training quantization* [113], [128] and *quantization-aware training* [123], [129]. However, both often require dequantiza-

tization during inference to handle mixed precision, adding latency and memory bandwidth overhead (*e.g.*, converting INT8 back to FP16 for matrix multiplication). To overcome this, T-MAC [114] employs bitwise operation lookup tables for direct low-bit computation, eliminating dequantization. Edge-LLM [130] further introduces adaptive quantization, feature caching, and value-density-based scheduling in a server-edge framework, improving utilization and inference speed.

4) Dynamic Knowledge Distillation: Dynamic knowledge distillation compresses FMs by transferring knowledge from heavy “teacher” models to lightweight “student” models. Key strategies include: *Adaptive temperature scaling*, which adjusts softmax temperatures to emphasize different levels of teacher knowledge [115], [131]; *Progressive transfer*, which gradually distills layer-wise knowledge to balance efficiency and accuracy [116], [124]; and *Dynamic weighting*, which reweights knowledge components according to task context [122], [125]. Representative methods such as TinyBERT [124], Patient KD [116], and MiniLM [131] demonstrate that distillation can significantly reduce model size and inference latency while retaining accuracy. By preserving multi-level contextual features, dynamic KD provides a scalable and adaptive solution for efficient long-context FM deployment.

D. Dynamic Routing

Dynamic routing enables *input-aware* allocation of computation at the *topological level*, determining which modules, layers, or branches are executed. Unlike dynamic attention that adapts feature-level computation, dynamic routing learns *conditional mappings* between inputs and network topology, allowing flexible path selection. This adaptivity reduces redundancy for simple inputs while preserving accuracy on complex ones, thus balancing speed, accuracy, and efficiency. We summarize five main paradigms, *i.e.*, *gate-based* [13], [132], [133], *affinity threshold-based* [42], [134], [135], *reinforcement learning-based* [43], [136], *early exiting* [58], [137]–[140], and *layer skipping* [141]–[143].

1) Gate-based Dynamic Routing with MoE: Gate-based dynamic routing enables *input-aware subnetwork activation* via differentiable gating (Fig. 12). In the *Mixture of Experts (MoE)* framework, a gating network assigns inputs across N experts by producing a probability vector $\mathbf{g}(\mathbf{x}) \in \mathbb{R}^N$ (often with

TABLE IV: Summary of dynamic routing techniques for MoE-based FM inference.

Categories	Technique highlight	Year	Ref
Dynamic routing (\$\S\$III-D)	Uses a load balancing loss to penalize imbalanced distributions.	2022	[13]
	Adjusts the gating function to be used to activate the next MoE module.	2024	[144]
	Only keeps key experts during runtime and dynamically maintain the swapping in and out of experts in memory.	2024	[81]
	Set a threshold as the basis for expert skipping and dynamically skip certain experts.	2024	[132]
	Dynamically schedule expert resources, predict each token's distribution in the next MoE layer by its expert path.	2023	[133]
	Introduces a learnable bias term for each expert, which is superimposed on the affinity scores.	2025	[42]
	Pipeline optimization and hierarchical loading strategies are achieved through affinity-awareness.	2024	[145]
	A bidirectional selection routing framework based on expert-token resonance achieves efficient routing.	2024	[134]
	Exploit inter-layer affinity in pre-trained MoEs to optimize placement and routing with one AlltoAll.	2024	[135]
	Model early exit as a reinforcement learning problem, use a "memory layer" to measure instance difficulty.	2024	[43]
Early exiting (\$\S\$III-D4)	Analyze key features to dynamically determine when to stop inference.	2024	[146]
	Shallow deep modules and synchronous parallel decoding are combined.	2023	[137]
	Uses entropy of internal representations to compute confidence for adaptive early exiting.	2022	[138]
	Scales early-exit LLMs to hundreds of billions of parameters with 3D parallelism.	2024	[139]
Layer skipping (\$\S\$III-D5)	Uses column-wise unified exits and monotonically decreasing exit layers.	2023	[140]
	Uses adaptive layer tuning with early exits/voting for memory-efficient full-model updates in edge LLMs.	2024	[58]
	Real speedups in small batches with one-shot depth pruning + pretraining, outperforming width pruning.	2024	[141]
	Merges later into earlier via parameter differencing, for training-free structure-pruning over 80% performance.	2024	[147]
Layer skipping (\$\S\$III-D5)	Learns token-wise skipping via binary router, for end-to-end training/inference, save FLOPs, boost few-shot performance.	2023	[142]
	Dynamically skips non-critical Transformer layers while evicting KV-cache, yielding plug-and-play 50% inference savings.	2024	[143]

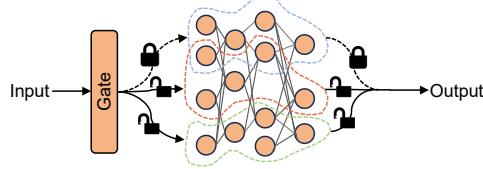
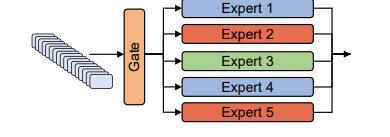


Fig. 12: Illustration of gate-based dynamic routing mechanism.

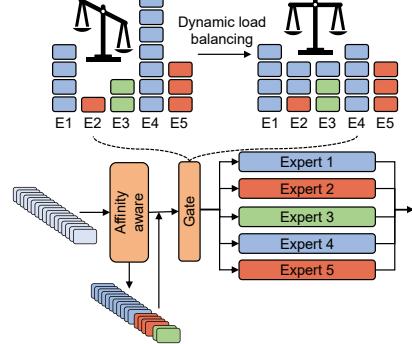
Gumbel-Softmax). A Top- K strategy (typically $K = 1$ or 2) activates only the selected experts, and the output is aggregated as $\mathbf{y} = \sum_{i \in \text{TopK}(\mathbf{g}(\mathbf{x}))} g_i(\mathbf{x}) \cdot E_i(\mathbf{x})$, where $E_i(\cdot)$ denotes expert i . This paradigm is widely adopted in large-scale models, such as Switch Transformer [13], which replaces FFN blocks with MoE for efficiency. However, MoE suffers from *expert load imbalance*, where a few experts are overused while others remain idle, causing memory overflow, degraded throughput, and poor hardware utilization. To address *load imbalance* in MoE, three strategies are commonly used: i) *load-balancing loss* [13], which penalizes uneven token routing; ii) *expert capacity constraints* [13], which cap tokens per expert with controlled dropping; and iii) *gate refinement* [81], [132], [133], [144], which improves routing quality. Representative designs combine these ideas. Switch Transformer [13] uses balancing loss with capacity buffering, while Pre-gated MoE [144] overlaps expert migration and execution to cut latency. Beyond balancing, *dynamic sparsity* methods skip low-contributing experts at runtime, as in SwapMoE [81] and expert pruning/skipping [132]. Lina [133] further schedules experts by popularity, enabling more adaptive resource allocation.

2) Affinity Threshold-based Dynamic Routing with MoE:

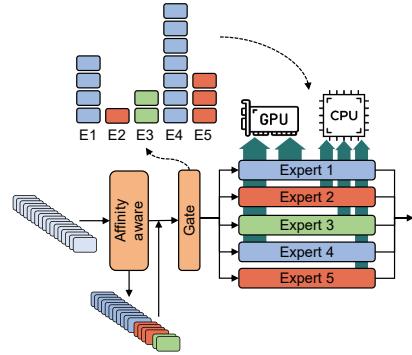
Affinity-based dynamic routing addresses the limitations of gating-based MoE, where fixed gating functions fail to adapt to dynamic input distributions and handcrafted balancing losses conflict with primary objectives, leading to overload, under-use, or instability [13]. Instead of static gating, it computes token-expert compatibility scores with learnable bias terms and real-time workload feedback [42], enabling adaptive, workload-aware routing without auxiliary losses (Fig. 13). Representative methods include DeepSeek-V3 [42], which employs sigmoid-based affinity scoring to improve utilization; Expert-Token Resonance MoE [134], which diversifies specialization via cosine similarity and an orthogonal GrAP layer;



(a) Dynamic routing with MoE.



(b) Expert-level load balancing.



(c) Device-level load balancing.

Fig. 13: Illustration of affinity threshold-based dynamic routing.

APTMoE [145], which reduces GPU memory and transfers by offloading sparse experts to CPUs; and Exflow [135], which co-locates stable cross-layer expert groups to cut communication latency. Collectively, these advances improve utilization, stability, and distributed efficiency for scalable agentic AI.

3) Reinforcement Learning-based Dynamic Routing with MoE: Reinforcement learning (RL)-based dynamic routing formulates expert selection in MoE as a *sequential decision-making task*. Unlike gating- or affinity-based approaches with fixed scoring, RL methods employ *policy networks* [136]

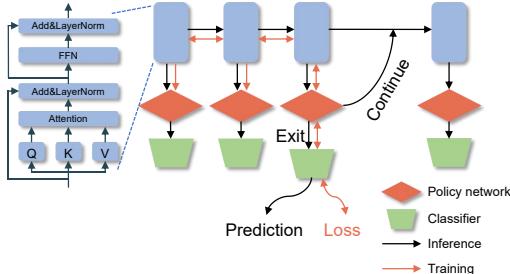


Fig. 14: RL-based dynamic routing.

and *reward-driven training* [43] to enable context-aware path selection and global multi-objective optimization. By adapting computation to input difficulty, they jointly optimize accuracy-latency trade-offs and mitigate local minima through policy exploration. ConsistentEE [43] exemplifies this paradigm, using RL to optimize early-exit policies (Sec. III-D4) via a memory layer for difficulty assessment and a difficulty-aware reward balancing prediction quality and latency. Policy gradients with multi-trajectory sampling further alleviate sparse rewards and align training with inference-time decisions (Fig. 14). Despite challenges in reward design and training stability, RL-based routing shows strong potential for flexible, task-adaptive inference in resource-constrained agentic AI.

4) *Early Exiting*: Early exiting accelerates inference by inserting adaptive decision points in shallow layers, allowing easy inputs to terminate once confidence is sufficient. This reduces latency and computation in FMs. Existing work mainly follows two directions, *i.e.*, *confidence-based exit strategies* [137], [138], [146] and *system-level integration* [58], [139], [140]. *First, confidence-based* methods focus on *when to exit*, balancing accuracy and efficiency through confidence measures. For example, AdaInfer [146] uses token-level features (*e.g.*, gap, top prob), FREE [137] applies a BMM-based estimator with parallel decoding, and CALM [138] evaluates softmax response, hidden state saturation, and classifier scoring. *Second, system-level* integration improves compatibility with optimizations like KV cache reuse and batch decoding. EE-LLM [139] overlaps KV computation and token generation via pipelined scheduling, SkipDecode [140] enforces monotonic exit depths to minimize recomputation, and Edge-LLM [58] introduces confidence-based voting with sensitivity-aware compression and hardware-aligned scheduling.

5) *Layer skipping*: Layer skipping accelerates inference by dynamically bypassing intermediate layers, unlike early exiting which terminates the entire sequence. It adaptively adjusts computational depth at the token or layer level, offering fine-grained acceleration. Research falls into three main directions. *i) Importance-based pruning*: Shortened LLaMA [141] ranks block importance via Taylor+ and PPL metrics to prune uncritical layers, while LaCo [147] merges adjacent layers to reduce depth without losing structure. *ii) Token-aware skipping*: SkipLayer [142] uses binary routing to decide per-token layer execution, and D-LLMs [143] combine decision modules with eviction policies to reduce compute and KV cache usage. *iii) Decoding-oriented skipping*: SkipDecode [140] skips shallow layers and reuses deeper computations during generation, while Draft-Verify accelerates speculative decoding by selec-

tively skipping intermediate layers.

E. Dynamic KV Cache Management

Dynamic Key-Value (KV) cache management underpins adaptive Transformer inference by reducing redundant computation in autoregressive decoding, lowering complexity from quadratic $O(n^2)$ to linear $O(n)$. However, memory grows linearly with sequence length, causing overflow and bandwidth-induced latency, which are bottlenecks for mobile/edge devices and real-time applications. To sustain responsiveness, recent work pursues adaptive cache management along three fronts.

1) *Context-Aware KV Cache Optimization*: Context-aware KV cache optimization dynamically regulates cache usage by prioritizing high-importance tokens and offloading low-value ones. Importance scores, usually derived from attention weights, guide three main strategies: *discarding* redundant tokens [44], [45], *merging/clustering* similar ones [117], [148], and *quantization* for compact storage [157]. Representative designs include H2O [44] and Scissorhands [45], which assess token relevance over time; PQ-Cache and RetrievalAttention [158], which employ similarity search (MIPS/ANNS) for cache selection; and SqueezedAttention [117] and ClusterKV [148], which cluster tokens to improve efficiency. At the storage level, InfLLM [149] hierarchically manages CPU-GPU memory, while Keyformer stabilizes eviction policies. Recent quantization methods (*e.g.*, KIVI, KVQuant, QServe, IntactKV) further compress key-value states to optimize memory and bandwidth.

2) *Attention-Aware KV Cache Optimization*: Core strategies include *KV sharing* (MQA [150], GQA [151]), which compress cache by sharing KV pairs across heads or groups; *latent compression* (MLA [42]), which replaces full KV with low-rank latent vectors; and *approximate or hybrid attention* (FLASH [40], Infini-Attention [152]), which combine gating, linear approximations, or local-global hybrids to balance efficiency and long-range modeling. For instance, MQA reduces cache size to $1/n_{\text{head}}$ but risks accuracy loss, while GQA offers a tunable trade-off via group size. MLA achieves compression without major degradation, and FLASH/Infini-Attention improve scalability by approximating or hybridizing attention computation.

3) *Model-adaptive KV Cache Optimization*: Model-adaptive KV cache optimization improves efficiency by tuning cache budgets to *layer*, *head*, and *model-specific* traits such as token/attention distributions, eviction-loss bounds, and retrieval ability. PrefixKV and DynamicKV [153] adjust allocation via prefix patterns and attention scores, while PyramidKV [154], PyramidInfer, and MEDA [159] exploit cross-layer heterogeneity with pyramid- or entropy-based quotas. AdaKV [155] bounds eviction loss for dynamic head budgets, and HeadKV [156] further prioritizes heads by retrieval and inference value. Together, they support elastic cache-compute trade-offs aligned with model structure.

F. Model-Adaptive System Scheduling

Model-Adaptive System Scheduling integrates algorithmic design with system-level scheduling to overcome FM inference bottlenecks by maximizing hardware utilization under

TABLE V: Summary of dynamic KV cache management techniques for elastic FM inference.

Categories	Technique highlight	Year	Ref
Dynamic KV cache management (\$III-E)	Context-aware KV cache optimization (\$III-E1)	Evaluates token importance via cumulative attention in sliding windows for critical tokens with long-term impact.	2023 [44]
		Focuses on sustained temporal influence of historical high-importance tokens for token relevance consistency.	2023 [45]
		Enhances cache efficiency via clustering-based token eviction, boosting accuracy and resource utilization.	2024 [148]
		Optimizes KV cache via CPU-GPU hierarchy, offloading less-accessed to CPU.	2024 [149]
	Attention-aware KV cache optimization (\$III-E2)	Accelerates incremental Transformer decoding by sharing key-value heads with only minor quality loss.	2019 [150]
		Enables efficient uptraining from MHA checkpoints with 5% pre-training compute for MHA-quality at MQA-speed.	2023 [151]
		Projects queries, keys, and values into a low-dimensional latent space for attention computation.	2024 [42]
		Compresses KV into a fixed associative memory and streams up to 1M tokens with local + linear attention.	2025 [152]
	Model-adaptive KV cache optimization (\$III-E3)	Dynamically assigns cache space via average attention scores to optimize layer-specific utilization and accuracy.	2025 [153]
		Leverages attention heterogeneity to allocate cache pyramidal, prioritizing more resources for lower layers.	2025 [154]
		Guides cache allocation by analyzing the theoretical upper bound of eviction loss for controllable resource management.	2025 [155]
		Implements attention-head-level cache allocation via dual capability evaluation and dynamic budget pooling.	2024 [156]

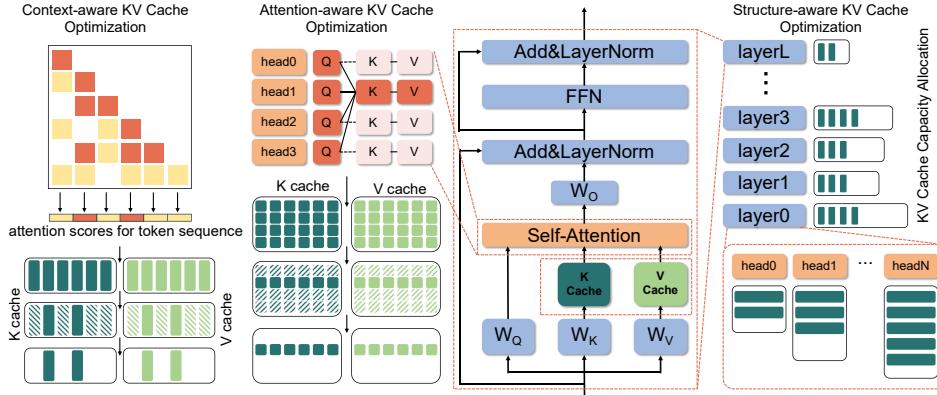


Fig. 15: Dynamic KV cache management.

TABLE VI: Summary of adaptive parallelism scheduling techniques for efficient FM inference.

Categories	Technique highlight	Year	Ref
Adaptive parallelism scheduling (\$III-F1)	Data parallelism (\$III-F1)	Boosts LLM inference throughput via OS-style paging (PagedAttention) for on-demand, shareable KV cache blocks.	2023 [163]
		Partitions parameters and KV caches across nodes to reduce memory footprint using full sharding.	2025 [164]
		Employs iteration-level scheduling to enable continuous batching.	2022 [165]
		Dynamically adjusts data processing granularity across heterogeneous nodes.	2025 [160]
	Sequence parallelism (\$III-F1)	Speeds up exact attention via tiling/recomputation, avoiding attention matrix storage and cutting GPU accesses.	2024 [59]
		Partitions sequences of up to one million tokens to achieve significant speedups.	2022 [166]
		Sequence partitioning, local/global attention optimization reduce communication/memory overhead in long sequences.	2024 [167]
	Pipeline parallelism (\$III-F1)	Boosts LLM speed via async pipelined speculation/early cancellation for low-acceptance/low-bandwidth scenarios.	2024 [168]
		Applies dynamic micro-batching for efficient multi-task inference.	2024 [79]
		Refines execution granularity through task scheduling to minimize pipeline stalls.	2023 [169]
		Refines execution granularity and manages memory via swapping to minimize stalls.	2025 [170]
	Expert parallelism (\$III-F1)	Improves cache hit rates through adaptive partitioning and VRAM budgeting.	2025 [161]
		Reformulates experts into block-sparse General Matrix Multiplications (GEMMs).	2025 [171]
		Implements dynamic load balancing using sparse activation of experts.	2022 [14]
	Heterogeneous processor parallelism (\$III-F1)	Leverages LLM power-law activation, GPU-CPU hybrid and sparse operators for fast, accurate consumer GPU inference.	2024 [172]
		DistServe disaggregates prefill/decoding, cuts interference, optimizes resources to boost GPU goodput.	2024 [79]
		HD-MoE optimizes MoE LLMs on 3D NMP via offline hybrid parallel mapping and online dynamic scheduling	2025 [160]
		Addresses LLM inference I/O bottlenecks on resource-constrained devices via CPU-GPU hetero-parallelism and async overlap	2024 [173]

dynamic workloads while preserving accuracy. It operates at two levels, the *front-end*, which optimizes abstract computation graphs through redundancy elimination, intermediate simplification, and parallelism strategies such as data, sequence, pipeline, and expert parallelism [160], [161]; and the *back-end*, which tunes execution to device capabilities via graph-level (e.g., operator fusion, graph rewriting [162]), memory-level (e.g., dynamic allocation, swapping), and instruction-level optimizations (e.g., loop unrolling, register tiling).

1) *Adaptive Parallelism Scheduling*: It dynamically adjusts execution strategies to system states (e.g., memory, compute, bandwidth) for efficient FM inference in resource-constrained agentic environments. Key paradigms include *data parallelism*, *sequence parallelism*, *pipeline parallelism*, *expert parallelism*, and *heterogeneous processor parallelism*. The main challenges are memory efficiency, real-time scheduling, and communication-aware partitioning, with adaptive strategies seeking to optimize throughput and latency while preserving accuracy for mobile/edge deployment (Tab. VI).

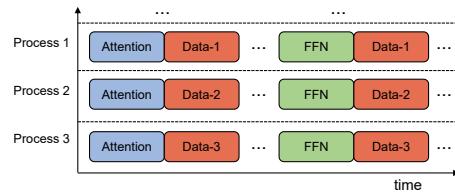


Fig. 16: Illustration of data parallelism. Each process performs the same operations on different data subsets simultaneously.

a. **Data parallelism**. It distributes data batches across devices to overcome memory limits and communication overhead (Fig. 16). Approaches include *full replication* (e.g., vLLM [163], TensorRT-LLM), which is fast but memory-hungry; *full sharding* (e.g., Seesaw [164]), which reduces footprint but increases communication; and *hybrid strategies* that balance both. Systems like Orca [165] apply iteration-level scheduling, while HD-MoE [160] adapts granularity across heterogeneous nodes. Recent advances explore *dynamic re-*

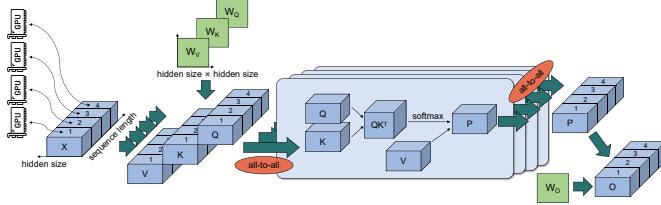


Fig. 17: Illustration of sequence parallelism.

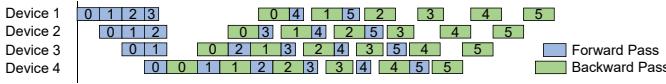


Fig. 18: Illustration of the compute efficiency challenges introduced by pipeline bubbles under 1F1B.

sharding to cut overhead and sustain efficiency under non-uniform bandwidth.

b. Sequence parallelism. It accelerates long-sequence inference by splitting inputs across devices, reducing compute and memory pressure (Fig. 17). Techniques span *distributed attention* [59], [60], [174], *sequence partitioning* [164], [166], *token-level balancing* [160], and *dynamic reshaping* [167]. Representative systems include DeepSpeed-Inference ($2.5\times$ speedup on million-token inputs), Sarathi ($10\times$ throughput via reshaping), StreamLLM (latency reduction via streaming), and RingAttention (near-unlimited context without memory overhead).

c. Pipeline parallelism. Pipeline parallelism accelerates FM inference by partitioning models into sequential stages across devices, improving memory utilization and efficiency for large models and long sequences. In dynamic or heterogeneous settings, adaptive scheduling addresses pipeline bubbles and latency bottlenecks. Strategies include *adaptive partitioning* for balanced workload and memory [79], [168], *task scheduling* to reduce stalls [169], and *memory management* via swapping [170], [172]. Representative systems such as PipeInfer [168], DistServe [79], SpecInfer [169], PowerInfer [172], and FlexInfer [170], demonstrate these optimizations through dynamic recomputation, micro-batching, and fine-grained scheduling.

d. Expert parallelism. It scales Mixture-of-Experts (MoE) models by activating only a subset of experts per input, reducing computation but introducing challenges of all-to-all communication, load imbalance, and dynamic scheduling. Experts are distributed across devices, requiring efficient cross-node coordination. Recent work advances along four fronts: *communication and scheduling* (Occult, SpeculativeMoE, HD-MoE [160]), *flexible execution* with adaptive partitioning and hardware-agnostic communication (MoEpic [161]), *block-sparse and pipelined execution* to overlap compute and communication (MoE-Lightning [171], MoE-Lens), and *dynamic load balancing* via sparse activation and cross-device routing (GLaM [14], Uni-MoE [175]).

e. Heterogeneous processor parallelism. Embedded devices typically integrate diverse processors (e.g., CPUs, GPUs, NPUs), creating opportunities to accelerate FM inference but also challenges in balancing loads and minimizing communication overhead. Recent systems (PowerInfer [172], Splitwise [176], DistServe [79], HD-MoE [160], FlexGen [177],

HeteGen [173]) dynamically map tasks to the most suitable processor. For example, PowerInfer exploits GPUs for large-scale sparse activations while offloading sequential I/O-heavy tasks to CPUs, boosting throughput and efficiency. Adaptive schedulers refine this mapping in real time based on processor capabilities and workload conditions, ensuring sustained performance and energy efficiency.

2) Computation Graph-level Optimization: Computation graph-level optimization restructures the overall FM graph—beyond operator-level tuning—to reduce redundant computation, memory access, and scheduling overhead. Two main approaches exist: *operator fusion*, which merges adjacent ops into composite operators to cut runtime and improve locality (e.g., tensor fusion [59], [60], [174], kernel fusion [12]; FlashAttention-2 [60], vLLM [163]); and *dynamic graph optimization*, which adapts graphs at runtime via on-demand construction, lazy evaluation, or rewriting (e.g., DyNet [178], TensorFlow Fold [179], Dali [180]). These methods jointly reduce compute/memory costs while preserving flexibility for deployment in constrained agentic systems.

3) Load Balance Scheduling: Efficient computation-to-device mapping is vital for agentic systems, as it directly impacts bandwidth, compute efficiency, and communication overhead. Existing work falls into two lines: *cross-device scheduling*, which mitigates skew and bottlenecks in distributed agent/edge environments [184]–[186] through methods such as tensor slicing with expert parallelism (DeepSpeed-MoE [181]) or expert-popularity prediction for dynamic scheduling (Lina [133]); and *heterogeneous chip mapping*, which balances workloads across CPUs, GPUs, NPUs, and DSPs with divergent compute and memory capacities [182], [183], exemplified by MoE-LightNING [171] via CGPIPE overlapping CPU/GPU compute and I/O to boost utilization.

4) Memory-level Optimization: Memory management is critical for FM performance in resource-limited agentic environments. To sustain responsiveness and scalability, while supporting efficient retraining, key strategies include *memory recomputation* [187]–[190], *partitioning optimization* [164], [185], [191], [192], and *diverted offloading* [193]–[195], which collectively reduce memory usage, balance compute–memory trade-offs, and adapt to runtime constraints (Tab. VIII).

a. Memory re-computation. It reduces peak usage by discarding and regenerating intermediate tensors, trading compute for memory. Strategies include *tensor rematerialization* [187], *selective recomputation* [188], and *adaptive partitioning* [189], [190]. InfiniGen [187] rematerializes KV tensors with *adaptive eviction*; KVPR [188] performs *I/O-aware* partial regeneration; Pie [189] mitigates CPU–GPU fragmentation via swapping; LLM-PQ [190] integrates *phase-aware* quantization with partitioned allocation; HybridCache combines checkpointing and hybrid caching for long-context case.

b. Memory partitioning optimization. This line distributes model states across hardware to reduce peak memory and communication bottlenecks. Techniques include *dynamic sharding* [164], [191], *deduplication* [192], and *hierarchical synchronization* [185]. Seesaw [164] and DynamoLLM [191] enable re-sharding for linear scaling; KV compression and

TABLE VII: Summary of computation graph-level optimization and load balancing scheduling for elastic FM inference.

Categories		Technique highlight	Year	Ref
Computation graph-level optimization (§III-F2)	Operator fusion (§III-F2)	Boosts attention efficiency via optimizing non-matmul FLOPs, sequence-parallelism, and intra-block warp partitioning for GPUs. Surveys LLM efficient inference via data/model/system-level optimizations, analyzes bottlenecks, with experiments.	2023 [60] 2023 [163]	
	Dynamic computation graph optimization (§III-F2)	Enables efficient LLM serving via KV cache paging (non-contiguous), cuts fragmentation, boosts throughput 2–4×. Dynamic computation graph declaration, optimizes construction overhead, supports dynamic structures, faster than peers.	2023 [163] 2017 [178]	
		Enables batched dynamic graph learning via dynamic batching, emulating dynamic graphs with static ones. Applies lazy compilation to dynamic computation graphs to boost ML system efficiency.	2017 [179] 2018 [180]	
Load balancing scheduling (§III-F3)	Cross-device load balancing scheduling (§III-F3)	Extends MoE to NLG, uses PR-MoE/MoS to shrink model, optimizes inference for speed/cost. Accelerates distributed MoE training and inference via targeted optimizations for efficiency.	2022 [181] 2023 [163]	
	Heterogeneous chip mapping (§III-F3)	Surveys CPU-GPU heterogeneous computing techniques across layers, covers systems/suites to boost performance/efficiency. Enables scalable, memory-efficient DNNs via virtualization techniques for memory management.	2017 [182] 2016 [183]	
		Uses CGOPipe and HRM to achieve high-throughput MoE inference on memory-constrained GPUs, outperforming existing systems.	2025 [171]	

TABLE VIII: Summary of memory-level optimization techniques for elastic FM inference.

Categories		Technique highlight	Year	Ref
Memory-level optimization (§III-F4)	Memory re-computation (§III-F4)	Speculates attention patterns to prefetch critical KV cache, cutting CPU-GPU transfer overhead.	2024 [187]	
		IO-aware partial KV cache recomputation enhances LLM inference efficiency.	2024 [188]	
		Leverages CPU memory via transparent swapping and adaptive expansion for efficient LLM inference.	2023 [189]	
	Memory partitioning optimization (§III-F4)	Uses phase-aware partition and adaptive quantization for efficient LLM serving on heterogeneous clusters.	2024 [190]	
		Dynamically adjusts parallelism across prefill/decode, with KV buffering/scheduling cutting overhead.	2025 [164]	
		Dynamically tunes instances, parallelism, frequency via hierarchical control for energy efficiency under SLOs.	2025 [191]	
	Memory diverted offloading (§III-F4)	Uses trained prompt tokens for parallel prediction, with hardware-aware sparse tree for efficiency.	2024 [192]	
		Conducts LLM inference limit study, focusing on bandwidth, sync, capacity via hardware-agnostic model.	2025 [185]	
		Offloads attention compute and KV cache to CPU via asymmetric pipelining, boosting throughput.	2024 [193]	
		Offloads KV cache to CPU via head-wise strategy with optimizations, cutting GPU memory.	2025 [194]	
		Employs CSDs for near-storage attention computation, with delayed writeback/X-cache cutting I/O to boost LLM throughput.	2025 [195]	

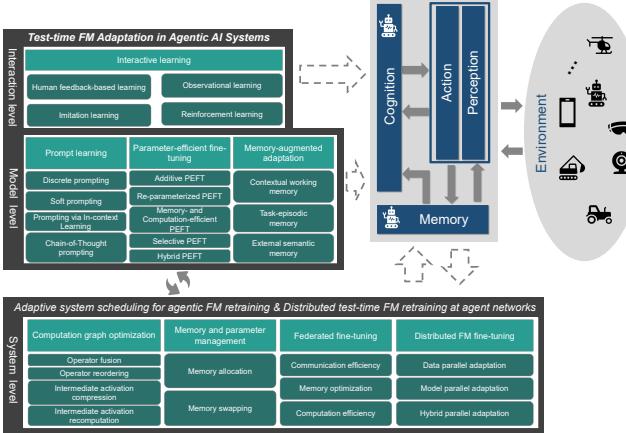


Fig. 19: Overview of test-time FM adaptation techniques.

hardware-aware decoding [192] cut redundancy; synchronization frameworks [185] reduce latency and balance loads.

c. Memory diverted offloading. This strategy shifts data or computation from GPUs to CPUs, NVMe, or near-storage accelerators to balance pressure, bandwidth, and latency. Approaches include *dynamic partitioning* [193], *real-time reallocation* [194], and *near-storage processing* [195]. Neo [193] redistributes tensors across CPU–GPU to cut transfers; HeadInfer [194] integrates swapping with scheduling; Aqua and INF² [195] exploit near-storage computing to minimize data movement.

IV. TEST-TIME FM ADAPTATION IN AGENTIC AI SYSTEMS

In dynamic open-world environments, agentic AI systems on platforms such as autonomous vehicles, drones, and service robots must continually adapt their FMs to evolving conditions (*e.g.*, traffic, lighting, user intent, novel stimuli). This requires *test-time adaptation*, *i.e.*, updating models during inference without full retraining and often without labeled data, to remain robust under distribution shifts, long-horizon tasks, and partial observability. Two tracks enable adaptive and efficient FM adaptation, *i.e.*, *algorithmic strategies* and *system-level techniques*. Beyond conventional gradient-based updates, FMs can also refine knowledge through *memory*, *external inte-*

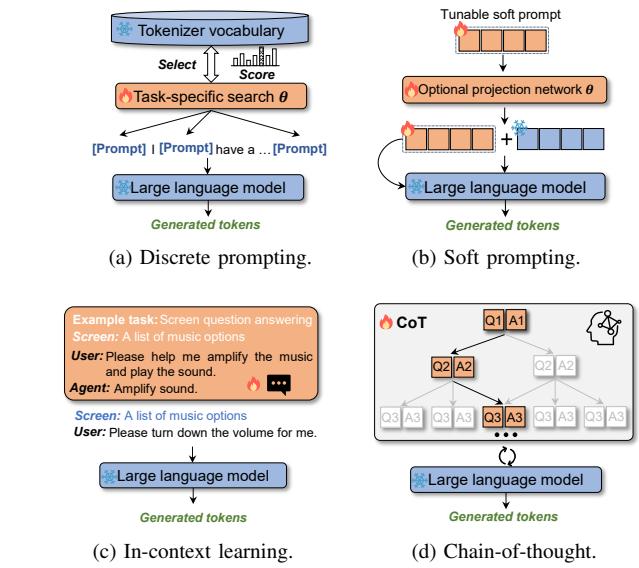


Fig. 20: Illustration of prompt tuning.

gration, and *prompt-driven control*. Algorithmic methods include prompt learning, parameter-efficient fine-tuning (PEFT), memory-augmented adaptation, and interactive learning, all designed to minimize parameters, samples, and overhead for rapid adaptation. In parallel, *system-level* techniques enhance adaptation under hardware limits via memory management, execution scheduling, and distributed adaptation.

A. Prompt Tuning

Prompt learning enables efficient test-time adaptation of FMs by steering behavior through *input sequence modification without parameter updates*, making it well-suited for resource-constrained AI agents. As illustrated in Fig. 20 and Tab. IX, it spans four strategies—*discrete prompting*, *soft prompting*, *in-context learning*, and *chain-of-thought prompting*, each balancing control granularity, model accessibility, and compute cost.

1) Discrete Prompting: Discrete prompting inserts optimized natural-language tokens into inputs to steer frozen FMs

TABLE IX: Summary of prompt tuning techniques for test-time FM adaptation in agentic AI systems.

Categories	Technique highlight for improving	Year	Ref
Prompt tuning (\$IV-A)	Discrete prompting (\$IV-A1)	Shared trigger tokens, top-k candidates, optimized prompt construction.	2020 [46]
		Maintain continuous embeddings, project to nearest vocabulary tokens.	2023 [196]
		Reinforcement learning for prompt optimization, policy network, z-score rewards.	2022 [197]
	Soft prompting (\$IV-A2)	Collaborative soft prompt training, learnable embeddings, aggregated updates.	2023 [48]
		Global/domain prompts, optimization, momentum aggregation, prompt similarity.	2024 [198]
	Prompting via in-context learning (\$IV-A3)	Task decomposition, sub-task examples, self-correction, human feedback.	2024 [47]
		Example selection with conditional DPP, capture relevance and diversity.	2023 [199]
		View hierarchies to structured text, chain-of-thought prompting.	2023 [200]
	Chain-of-Thought (CoT) prompting (\$IV-A4)	Five-stage CoT, expand concepts, continuation and revision modules, lightweight adapters.	2024 [201]
		Attention saliency for adaptive CoT prompt selection, zero-shot reasoning.	2024 [202]
		Integrate language, perception, control in multimodal Transformer, four-stage reasoning.	2024 [203]
		Multi-hop rationales, commonsense relations, dual-stage alignment.	2023 [204]

without parameter updates (Fig. 20a), offering black-box compatibility and negligible overhead. Methods include *gradient-based* approaches, such as AutoPrompt [46] (gradient-guided token search) and PEZ [196] (embedding-space optimization projected to tokens), as well as *RL-based* methods like RLPrompt [197], which frames prompting as reinforcement learning with reward shaping and input-conditioned generation for stronger few-shot generalization.

2) *Soft Prompting*: Soft prompting prepends *trainable continuous embeddings* to inputs, offering finer control than discrete tokens while keeping FM weights frozen (Fig. 20b). PROMPTFL [48] applies this to federated CLIP via learnable embeddings prepended to text tokens. DiPrompT [198] extends to federated domain generalization with disentangled prompts. A global prompt for invariant knowledge, local prompts optimized via prototypical guidance and momentum aggregation, and query prompts for label-free domain inference, ensembled at inference for cross-domain generalization.

3) *Prompting via In-context Learning (ICL)*: ICL adapts FMs *parameter-free* by prepending task-specific exemplars, conditioning inference on recent context (Fig. 20c). This is especially valuable for agents with limited resources and strict latency, enabling dynamic adjustment from signals such as user interaction or episodic memory. Applications include Wang *et al.* [200], which reformulates Android UI trees into HTML-style text for summarization and QA; MobileGPT [47], which enhances generalization via hierarchical memory, dynamic prompt reconstruction, and human-in-the-loop refinement; and CEIL [199], which optimizes exemplar selection with conditional DPPs for relevance–diversity balance, yielding strong transferability.

4) *Chain-of-Thought (CoT) Prompting*: Chain-of-thought (CoT) prompting augments inputs with natural language rationales, guiding models through intermediate reasoning steps before producing answers (Fig. 20d). Although it increases inference length, CoT improves multi-step decision-making, semantic planning, and interpretability, making it valuable under mobile resource constraints. Recent work highlights *dynamic adaptation*, where CoT adjusts reasoning strategies to context and distribution shifts [202], [204]. Instance-adaptive Zero-shot CoT [202] selects prompts via attention-based saliency with substitution or voting, enabling task-agnostic adaptation. DOCTOR [204] builds multi-hop rationales through iterative QA with commonsense guidance and dual-stage alignment for coherence. ECoT [203] integrates language, perception, and control in a multimodal Transformer, supporting test-time adaptation via meta-learning. PromptCoT [201] extends CoT

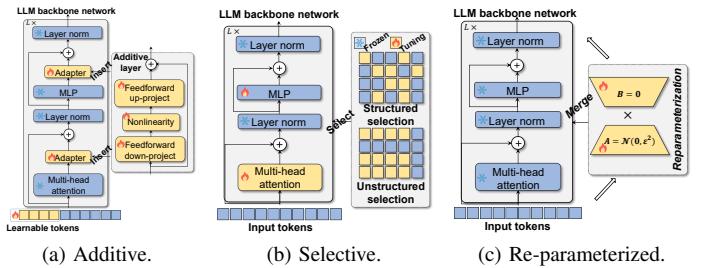


Fig. 21: Illustration of parameter-efficient fine-tuning.

to diffusion models with staged reasoning and lightweight adapters for efficient multi-task adaptation.

B. Parameter-efficient Fine-Tuning (PEFT)

On mobile and edge platforms, full-parameter fine-tuning is infeasible due to prohibitive memory, compute, and storage demands. Parameter-Efficient Fine-Tuning (PEFT) alleviates this by updating only *small, strategically chosen parameter subsets* while keeping the backbone frozen (Fig. 21). This enables rapid, lightweight, and even on-device adaptation, supporting heterogeneous, personalized, and real-time applications. PEFT methods can be grouped into following four categories.

1) *Additive PEFT*: Additive PEFT facilitates efficient adaptation of FMs under resource constraints by *injecting* lightweight trainable modules into Transformer layers or input embeddings, while keeping the backbone parameters frozen. It reduces training-time memory and computation overhead by updating only a small number of the inserted modules. Also, it avoids gradient storage and optimizer tracking for the frozen backbone and limits activation retention to the trainable components, resulting in efficient fine-tuning under memory and energy constraints. Depending on the insertion location and architectural design, additive PEFT methods are commonly categorized into adapter tuning [205]–[207], prompt tuning [210]–[212], and other integration-based approaches [213], [214].

a. *Adapter tuning*. Adapter tuning adapts FMs by inserting lightweight modules into Transformer layers while freezing the backbone, enabling efficient and task-specific customization. Recent advances improve efficiency, modularity, and on-device suitability. SparseAdapter [205] prunes adapter weights at initialization to scale capacity under fixed budgets. Comacter [206] applies low-rank hypercomplex decomposition, reducing complexity from $O(kd)$ to $O(k + d)$. LiteMoE [207] extracts lightweight proxy submodels from MoE-based LLMs

TABLE X: Summary of Parameter-efficient fine-tuning for test-time FM adaptation in agentic AI systems.

Categories		Technique highlight for improving	Year	Ref
Parameter-efficient fine-tuning (PEFT) (§IV-B)	Additive PEFT (§IV-B1)	Prune adapter weights at initialization, introduce Large-Sparse configuration, boost capacity under parameter budget.	2022	[205]
		Introduce low-rank hypercomplex adapters, compute task-specific weights with Kronecker products, reduce parameter complexity.	2021	[206]
		Extract lightweight proxy submodels, identify and merge important experts, enable efficient on-device adapter tuning.	2024	[207]
		Reformulate PEFT, pruning, and quantization as adapter-based transformations, enable consistent chaining of modules.	2024	[208]
	Selective PEFT (§IV-B1)	Use zeroth-order tensor-train adapters, apply parallel contraction and sublinear query scheduling, enable efficient fine-tuning.	2024	[209]
		Prepend continuous prefix vectors, reparameterize via MLP, enable stable training with frozen backbone.	2021	[210]
		Insert continuous prompts in all Transformer layers, use reparameterization encoders, improve parameter efficiency.	2021	[211]
	Other modules (§IV-B1)	Combine sample selection and noise-aware training, use in-memory computing for scaled retrieval, enable edge tuning.	2024	[212]
		Introduce learnable scaling vectors, rescale attention and feedforward activations, support mixed-task fine-tuning.	2022	[213]
	Selective PEFT (§IV-B1)	Train lightweight policy adapters, shape output distributions toward user objectives, combine adapters with base model.	2023	[214]
		Compute sensitivity scores for bias pruning, prune low-sensitivity biases and reinitialize important ones.	2023	[215]
		Update child network only, mask gradients of non-child parameters, preserve full model capacity.	2021	[216]
		Select parameters with largest absolute differences, retrain with binary masks and L1 regularization.	2021	[217]
		Filter smallest-magnitude parameters with group-wise selection, enable efficient non-IID adaptation.	2023	[218]
		Identify node-level importance with L1-norm changes, select top-r% nodes for learning.	2022	[219]
Re-parameterized PEFT (§IV-B3)	Unstructured selection (§IV-B1)	Update selected rows and columns in weight matrices, perform in-place fine-tuning.	2024	[220]
		Approximate weight updates with low-rank matrices, update only rank-constrained modules.	2022	[50]
		Sample dynamic target rank, truncate LoRA projection matrices, support flexible inference.	2022	[221]
	LoRA family (§IV-B3)	Parameterize updates with SVD-like decomposition, prune singular values based on importance.	2023	[222]
		Parallelize zeroth-order gradient estimation, update only one matrix in LoRA.	2024	[223]
		Reuse LoRA weights for compressed models, optimize recovery modules for degraded weights.	2023	[224]
		Quantize weight deltas to 1-bit sign representations, use trainable scale factors.	2024	[225]
	LoRA variants (§IV-B3)	Represent updates in frequency domain with DFT, learn shared spectral coefficients.	2024	[226]
		Perform SVD to extract principal component subspace, constrain updates within singular vectors.	2024	[227]
	Hybrid PEFT (§IV-B4)	Decompose PEFT design into layer grouping and strategy assignment, refine across backbones.	2023	[228]
		Search over insertion layers and module combinations, optimize parameter budgets, use Bayesian optimization.	2024	[229]

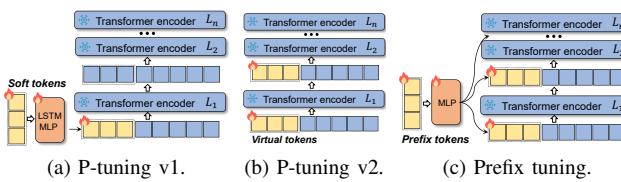


Fig. 22: Illustration of prompt tuning.

for multi-task personalization. AdaZeta [209] introduces tensorized forward-only adapters with adaptive scheduling for memory-efficient tuning. CLAM [208] integrates PEFT, pruning, and quantization into unified adapter-based transformations for consistent specialization and compression.

b. Prompt tuning. Prompt tuning adapts FMs by optimizing a small set of task-specific embeddings while freezing the backbone (Fig. 22), offering lightweight adaptation with minimal parameters and strong data efficiency. Structure-aware variants refine *injection position and form* within Transformers. Prefix-Tuning [210] prepends trainable prefix embeddings across layers via MLP reparameterization; P-Tuning [211] learns continuous embeddings with an LSTM encoder at the input, and P-Tuning v2 extends this to all layers for greater expressivity. NVCiM-PT [212] further tailors prompt tuning to edge devices through hardware-aware encoding, sample selection, and non-volatile memory optimization, enabling efficient and robust on-device adaptation.

c. Other structurally integrated modules. Beyond adapters and prompts, structurally integrated PEFT modules inject lightweight transformations into frozen backbones via *scaling*, *policy control*, or *distribution reshaping*, enabling multi-task and inference-time adaptation with negligible cost. (IA)³ [213] applies learnable task-specific scaling vectors to attention and feedforward activations, supporting mixed-task batches with minimal overhead, while IPA [214] leverages a lightweight policy adapter trained via reinforcement learning and integrated through a product-of-experts mechanism to steer distributions efficiently across diverse objectives.

2) Selective PEFT: Unlike additive PEFT, which inserts auxiliary modules, selective PEFT updates only a task-relevant subset of native parameters to reduce overhead while maintaining adaptability (Fig. 23). Approaches fall into two categories: *First*, *structured selection* updates larger architectural units

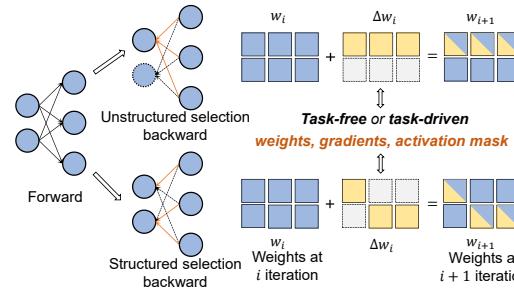


Fig. 23: Illustration of selective parameter-efficient fine-tuning.

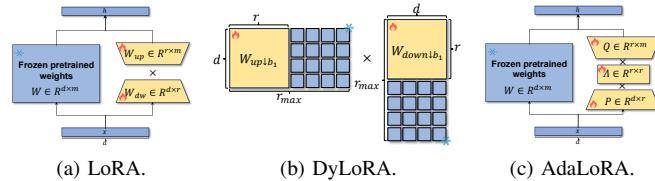


Fig. 24: Illustration of LoRA Family.

(e.g., layers, blocks, attention heads) for coarse-grained yet efficient tuning. FAR [219] ranks FFN nodes by $L1$ -norm changes and fine-tunes the top- $r\%$, reconfiguring memory layout to reduce fragmentation, while RoCoFT [220] restricts updates to selected rows/columns of weight matrices, achieving accuracy comparable to full fine-tuning. *Second, unstructured selection* targets individual parameters for maximal efficiency. U-BitFit [215] prunes low-sensitivity biases via gradient signals; CHILD-Tuning [216] masks gradients to update only a task-aware “child network”; LT-SFT [217] leverages lottery ticket sparsity to retrain selected weights; PaFi [218] tunes merely 0.5% of parameters chosen by group-wise magnitude, enabling efficient adaptation even in federated settings.

3) Re-parameterized PEFT (LoRA Family): It reformulates parameter updates via low-rank decomposition, training only rank-constrained matrices while freezing the backbone. This achieves lightweight, memory-efficient adaptation without inference overhead. The *LoRA family* is the canonical design. Given $\mathbf{W}_0 \in \mathbb{R}^{d \times m}$, updates are expressed as In Low-Rank Adaptation (LoRA) [50], the update to a weight matrix $\mathbf{W}_0 \in \mathbb{R}^{d \times m}$ is re-parameterized as

$$\mathbf{W} = \mathbf{W}_0 + \Delta\mathbf{W} = \mathbf{W}_0 + \alpha \cdot \mathbf{AB},$$

TABLE XI: Summary of memory- and computation-efficient PEFT for FM adaptation in agentic AI systems.

Categories	Technique highlight for improving	Year	Ref
Memory- and computation-efficient PEFT (§IV-B5)	Pruning-enhanced PEFT (§IV-B5)	Structured pruning with LoRA gradients: iterative LoRA pruning/fine-tuning, hardware-friendly sparsity	2023 [232]
		Prune redundant adapter params at init, scale capacity with bottleneck-sparsity balance, keep param efficiency	2022 [205]
		Score adapter salience, adjust adapter ranks with layer importance, use self-distillation.	2024 [233]
		Deploy client-specific LoRA modules, perform rank self-pruning, aggregate with sparsity weighting.	2024 [234]
	Quantization-aware PEFT (§IV-B5)	Quantize weights to 4-bit NF4, apply double quantization, offload optimizer states with paged optimizer.	2023 [129]
		Quantize pretrained weights, keep merged weights quantized for inference.	2024 [235]
		Quantize weights and LoRA adapters to FP8, use gradient scaling + fuse operators.	2024 [236]
		Preserve weak columns in FP16, group scaling, fine-tune sensitive columns.	2024 [237]
	Backpropagation-free PEFT (§IV-B5)	Generate task-specific parameters with hypernetworks, adapt without backpropagation.	2023 [238]
		Avoid backpropagation through backbone, leverage ladder side network with pruning and layer dropping.	2022 [239]
		Zero-order gradient estimation with two forward passes, skip backpropagation.	2023 [240]
		Project gradients into low-rank subspace, leverage gradient matrix structure.	2024 [241]
		Combine pruning and quantization for unified compression, tune layers adaptively with hardware scheduling.	2024 [58]

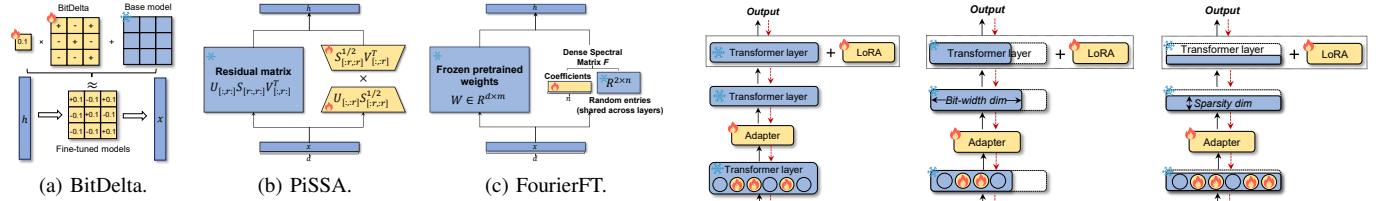


Fig. 25: Illustration of LoRA-inspired variants.

where $\mathbf{A} \in \mathbb{R}^{d \times r}$, $\mathbf{B} \in \mathbb{R}^{r \times m}$, $r \ll \min(d, m)$. Only \mathbf{A}, \mathbf{B} are trained, cutting parameter and memory costs substantially [50]. Recent extensions enhance LoRA with *dynamic rank adjustment*, *budget allocation*, *compression recovery*, and *gradient-free tuning*. DyLoRA [221] samples target ranks to avoid exhaustive search; AdaLoRA [222] prunes singular values for adaptive rank budgeting; CA-LoRA restores compressed LLMs via knowledge inheritance; Delta-LoRA [230] jointly updates pretrained weights and low-rank deltas; P-RGE [223] enables forward-pass-only tuning via zeroth-order estimation. LoRA-inspired variants introduce more flexible low-rank structures. BitDelta [225] compresses weight differences into 1-bit signs with scaling factors for multi-tenant deployment. FourierFT [226] encodes updates as sparse Fourier signals, achieving LoRA-level accuracy with $\leq 0.1\%$ parameters. PiSSA [227] initializes updates with top singular vectors from SVD, improving convergence and quantization robustness.

4) *Hybrid PEFT*: Hybrid PEFT combines multiple strategies (*e.g.*, LoRA, Adapters, Prompt Tuning) to leverage complementary strengths in efficiency, flexibility, and generalization, making it well-suited for resource-constrained agent scenarios [231]. S4 [228] formalizes hybrid design spaces along four axes—layer grouping, parameter allocation, group selection, and strategy assignment—identifying robust patterns such as spindle-shaped grouping and diverse strategy assignment across layers. AUTOPEFT [229] automates hybrid design via hierarchical search and multi-objective Bayesian optimization, exploring insertion layers, parameter budgets, and module combinations (*e.g.*, serial/parallel adapters with prefix-tuning) to yield Pareto-optimal and transferable configurations.

5) *Memory- and Computation-efficient PEFT*: In mobile/edge agent scenarios, FMs must adapt under tight memory, computation, and latency budgets. Conventional PEFT can incur high delays and memory peaks, particularly during backpropagation when activations, gradients, and optimizer states coexist. To address this, memory- and computation-efficient PEFT strategies (Fig. 26, Tab. XI) have emerged: *pruning-enhanced PEFT* [205], [232]–[234] trims redundant param-

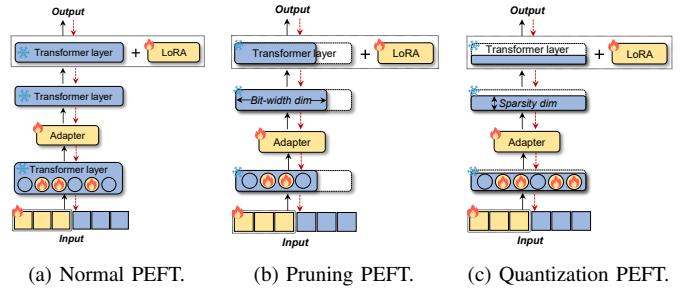


Fig. 26: Illustration of efficient PEFT design.

eters or adapter weights to cut training cost while retaining adaptation capacity; *quantization-aware PEFT* [129], [236], [237] compresses trainable modules into low-bit formats, balancing accuracy with memory and bandwidth efficiency; *backpropagation-free PEFT* [238]–[241] eliminates gradients via optimizer-free updates, forward-only optimization, or low-rank projection, lowering memory overhead and enabling scalable on-device tuning.

a. **Pruning-enhanced PEFT**. This approach combines parameter pruning with PEFT modules to boost sparsity and efficiency, supporting lightweight deployment and adaptive retraining. LoRAPrune [232] exploits LoRA’s low-rank structure for gradient-based structured pruning without full-model updates; APT [233] adaptively prunes adapters via outlier-aware salience scoring and dynamic rank adjustment; SparseAdapter [205] prunes at initialization to construct sparse yet expressive adapters; HETLoRA [234] integrates rank self-pruning with sparsity-weighted aggregation to enable client-specific adaptation in federated settings.

b. **Quantization-aware PEFT**. Quantization-aware PEFT reduces memory and compute by applying low-bit quantization to pretrained weights while fine-tuning selected parameters, enabling hardware-friendly, accuracy-preserving adaptation. QLoRA [129] introduces 4-bit NF4 quantization with double quantization and paged optimizers for long-sequence training; QA-LoRA [235] adopts group-wise quantization to preserve adaptation freedom under GPU limits; 8-bit Transformer [236] quantizes both backbone and LoRA adapters to FP8/Posit8 with fused operators for stable 8-bit training; QEFT [237] selectively retains weak FP16 columns while quantizing others, balancing efficiency with accuracy.

c. **Backpropagation-free PEFT**. This paradigm alleviates memory bottlenecks and optimizer overhead by removing or approximating gradient computation (Fig. 27). LST [239] introduces a ladder side network to bypass backbone backprop-

TABLE XII: Summary of memory-augmented adaptation for real-time FM adaptation on agentic AI systems.

Categories		Technique highlight for improving	Year	Ref
Memory-augmented adaptation (§IV-B5)	Contextual working memory (§IV-C1)	Prompt adaptation (§IV-C1) Modify Transformer attention masks, compress prompts into gist tokens, enable zero-shot gist prefix prediction.	2023	[51]
		Apply dynamic prompt compression, use iterative token-level compression, align with instruction fine-tuning.	2023	[242]
		Model prompt optimization as MDP, integrate error feedback via MCTS, refine prompts iteratively.	2023	[52]
		Divide long contexts into parallel windows, reuse positional embeddings, restrict attention to within-window tokens.	2024	[243]
		Model example selection as MDP, use marginal utility rewards, improve generalization across models.	2022	[53]
	Role playing (§IV-C1)	Adopt fact-grounded scene simulation, reconstruct experience pipeline, forget irrelevant knowledge for consistency.	2023	[244]
		Use orchestrator with task and progress ledgers, assign tasks and monitor outcomes, enable collaborative agent reasoning.	2024	[245]
	Self correction (§IV-C1)	Generate self-feedback for output refinement, act as generator, refiner, and feedback provider.	2023	[105]
		Assess confidence of model outputs, enable adaptive self-correction.	2024	[246]
	Task-episodic memory (§IV-C2)	Interact with external tools for validation, generate actionable feedback with LLM.	2023	[247]
		Combine embedding entropy and domain scores, build data buffer with diversity, replay dialogues for adaptation.	2024	[248]
		Cluster past data for replay, model replay as multi-armed bandit, select data to mitigate forgetting.	2024	[249]
	External semantic memory (§IV-C3)	Store self-experiences as triplet knowledge, automate retrieval with fuzzy matching.	2023	[250]
		Use SQL databases as symbolic memory, dynamically generate SQL commands.	2023	[251]
		Encode corpus knowledge into model parameters, use pseudo query-document pairs.	2022	[252]
		Build on-the-fly memory as task-episodic memory, maintain conversation consistency.	2023	[253]
	Continual knowledge graph learning (§IV-C3)	Construct domain knowledge graphs with LLMs, align with KG feedback for updates, address domain gaps.	2024	[254]
		Use evidence graph mining with LLMs, aggregate evidence graphs, enable graph-of-thoughts inference.	2023	[255]
		Extract triples for KG construction, explore nodes and relationships, enable multi-hop KGQA.	2024	[256]
	Continual document learning (§IV-C3)	Use indexing APIs for document-level updates, skip unchanged blocks, avoid redundant updates.	2024	[257]
		Manage document storage with incremental updates, index and parse new data.	2023	[258]

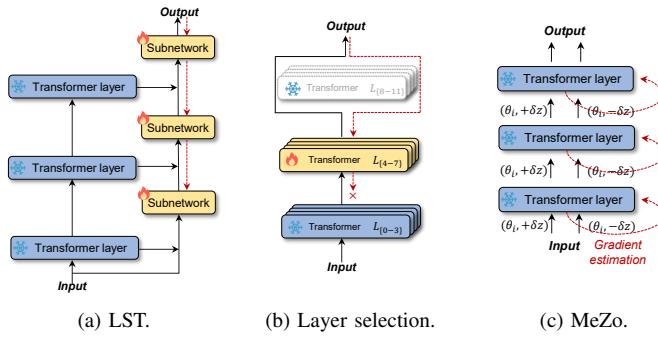


Fig. 27: Illustration of memory-efficient PEFT.

agation, cutting activation storage while retaining adaptation. HyperTuning [238] employs a hypernetwork to generate task-specific parameters (*e.g.*, prompts, LoRA) without gradients, lowering memory and compute costs. MeZO [240] applies zeroth-order optimization, estimating gradients via forward passes only, making adaptation memory usage close to inference. GaLore [241] projects gradients into low-rank subspaces, reducing optimizer state memory while preserving flexibility for both full and parameter-efficient tuning.

C. Memory-Augmented Adaptation

Unlike PEFT, which adapts parameters, memory-augmented adaptation equips FMs with external or auxiliary memory for recording, updating, and recalling task-relevant information during inference, enabling flexible test-time adaptation with minimal parameter updates (Tab. XII).

1) *Contextual Working Memory*: Agentic FMs on embedded devices often forget instructions or misinterpret follow-ups due to limited context windows and lack of persistent state. Contextual working memory provides short-term, task-specific storage of recent inputs, observations, and intermediate results (Fig. 28a), active only within an ongoing interaction and cleared on context shift. It supports coherent inference and adaptive responses via mechanisms such as prompt adaptation [51], [52], [242], long-context distillation [53], [243], role playing [244], [245], and self-correction [105], [247].

a. *Prompt adaptation*. Prompt adaptation compresses and refines historical interactions and instructions within limited context windows, enhancing inference consistency without parameter updates. Strategies include *soft compression* [51],

which distills prompts into gist tokens, *hard compression* [242], which prunes redundant tokens with dynamic ratio allocation, and *optimization* [52], which refines prompt structures via self-reflective feedback.

b. *Long-context distillation*. Long-context distillation extracts the most relevant spans from lengthy inputs to fit limited prompt windows. Methods include *context pruning* [243], which models example selection as an RL-based MDP, and *context fusion* [53], which aggregates information via parallel context windows with restricted attention and shared task tokens.

c. *Role playing*. Role playing steers inference and interaction by assigning agents task-specific identities. *Single-agent role playing* [244] simulates roles (*e.g.*, planner, explainer) to improve reasoning style and reduce hallucinations, while *multi-agent role playing* [245] orchestrates collaboration with distinct identities and task ledgers to support coordinated reflection and plan revision.

d. *Self correction*. Self-correction refines outputs at inference via feedback or internal evaluation, improving reliability without parameter updates, crucial for dynamic agentic environments. Approaches include *feedback-based refinement* [105], where SELF-REFINE iteratively critiques and revises outputs within a single LLM; *confidence-based adjustment* [246], where IoE triggers retries only when confidence is low to avoid over-correction; and *tool-augmented correction* [247], where CRITIC integrates calculators, search, or interpreters to drive “verify–correct” cycles.

2) *Task-Episodic Memory*: In long-horizon agentic tasks, recalling past actions, successes, and failures is essential to avoid redundancy and inefficiency. *Episodic memory* extends beyond short-lived working memory by storing structured records of actions, observations, and outcomes (Fig. 28b), enabling retrospective inference and experience reuse under sparse feedback. Approaches fall into two categories: *data replay*, which reuses logged interactions or embeddings to improve sample efficiency and mitigate forgetting, *e.g.*, diverse dialogue replay for personalization [248] or bandit-based clustering for dynamic sampling [249]; and *self experiences*, where agents generate and store semantic triplets, logs, or documents for introspection and decision-making, as in RET-LLM [250], ChatDB [251], CorpusBrain [252], and MemoChat [253].

3) *External Semantic Memory*: In open-ended environments, pretrained FMs alone cannot ensure reliable infer-

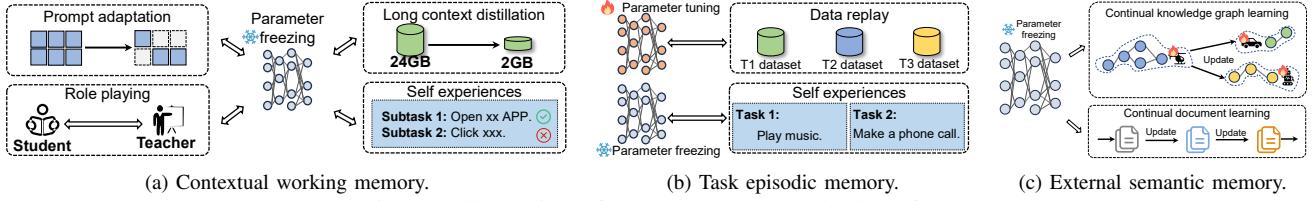


Fig. 28: Illustration of memory-augmented adaptation.

TABLE XIII: Summary of interactive learning for real-time FM adaptation on AI agents.

Categories	Technique highlight	Year	Ref
Interactive learning (\$IV-D)	Human feedback-based learning (\$IV-D1)	Simulate user interaction, heuristic feedback, CoT-based reasoning.	2024 [33]
		GPT-4 for fine-grained ratings, human preferences alignment.	2023 [259]
		LLM verifiers, corrective feedback, refine decision-making.	2025 [260]
		LLM inference for reward modeling, StableReinforce algorithm.	2025 [261]
		Bradley-Terry preference modeling, optimize policy via binary cross-entropy.	2023 [262]
		Reward model on human rankings, PPO for policy alignment.	2022 [54]
	Imitation learning (\$IV-D2)	Inverse soft Q-learning, occupancy and token likelihood, principled imitation.	2024 [263]
		Joint multimodal tokens, Swin Transformer, PPO for prompt tuning.	2024 [264]
		Align VLM and LLM expert with DAgger-DPO, distill expert actions and feedback.	2024 [55]
	Observational learning (\$IV-D3)	Bootstrap LLM planner with demonstration, iterative self-training, positive feedback.	2024 [265]
Reinforcement learning (\$IV-D4)	Observational learning (\$IV-D3)	Visual to textual prompts, CoT-based reasoning, in-context learning.	2024 [266]
		Cycle observation, action, and reflection, extract high-similarity subgraphs, LLM and KG synergy.	2024 [267]
		Pioneer-observer LLMs, alternating roles, shared rewards, policy co-adaptation.	2024 [268]
	Reinforcement learning (\$IV-D4)	Self-reflective feedback, layered memories, refine decision-making.	2023 [93]
		External experience memory with Q-learning, refine long-term memory via RL.	2023 [269]

ence; agents need access to factual knowledge, task schemas, and up-to-date information to avoid hallucination and improve generalization. *Semantic memory* provides persistent external representations, e.g., knowledge graphs, document stores, or vector databases, that complement model parameters and differ from episodic memory by encoding generalized facts and concepts (Fig. 28c). Two main approaches dominate. *First, continual knowledge graph learning* incrementally expands structured knowledge with new entities and relations, supporting consistent reasoning in dynamic domains, as in domain-specific alignment [254], evidence-graph inference (MindMap) [255], and KG-RAG triple construction for fine-grained reasoning [256]. *Second, continual document learning* ingests, summarizes, and indexes new documents for retrieval-based inference, using strategies such as block-skipping (LangChain [257]) and incremental document-aware storage (LlamaIndex [258]).

D. Interactive Learning

Despite advances in PEFT and memory mechanisms, AI agents on mobile/edge platforms still struggle with dynamic environments, partial observability, sparse feedback, and unpredictable dynamics, where offline training or static policies fall short. *Interactive learning* addresses this by refining agent behavior at test time through continuous interaction with environments, humans, or other agents (Tab. XIII). By leveraging real-time feedback (e.g., user corrections, failure signals, environmental changes), agents adapt efficiently without dense supervision or full retraining, enabling robust generalization in tasks like wearable user modeling, AR interaction, and navigation in novel environments. Interactive learning methods can be grouped into four categories, i.e., *Human feedback-based learning* [33], [261], *Imitation learning* [263], [265],

Observational learning [266]–[268], and *Reinforcement learning* [93], [270], [272].

1) *Human Feedback-based Learning*: Human feedback-based learning refines agent policies using lightweight human input (e.g., preferences, ratings, corrections), avoiding explicit reward design or full demonstrations. It is particularly effective when objectives are ambiguous, demonstrations unavailable, or task demands evolve, enabling flexible test-time adaptation (Fig. 29a). Feedback typically comes in three forms: *i) Preference feedback*, where humans select preferred behaviors (e.g., SimUser [33] generates usability preferences via CoT reasoning). *ii) Scalar feedback*, where behaviors receive numerical ratings (e.g., ULTRAFEEDBACK [259] provides fine-grained GPT-4 ratings for reward models). *iii) Corrective feedback*, where humans highlight and revise errors (e.g., V-Droid [260] employs LLM verifiers for real-time GUI correction). Three main training paradigms are used: *i) Reward modeling*, which learns explicit estimators from feedback (e.g., R1-Reward [261] reformulates multimodal reward modeling with StableReinforce). *ii) Direct Preference Optimization (DPO)* [262], which bypasses reward models by directly optimizing policies with binary preference probabilities. *iii) Reinforcement Learning with Human Feedback (RLHF)*, which integrates reward models into RL (e.g., InstructGPT [54]).

2) *Imitation Learning*: In agentic systems, designing explicit reward functions is often infeasible due to sparse feedback and limited resources. Imitation learning (IL) bypasses this by learning policies directly from expert demonstrations, where state-action trajectories encode task objectives implicitly. By mimicking expert behaviors, IL reduces reward-engineering overhead and accelerates deployment, making it effective for real-time agents such as drones, vehicles, and wearables (Fig. 29b). Recent advances adapt IL to LLM-based and multimodal agents: Geist *et al.* [263] introduce

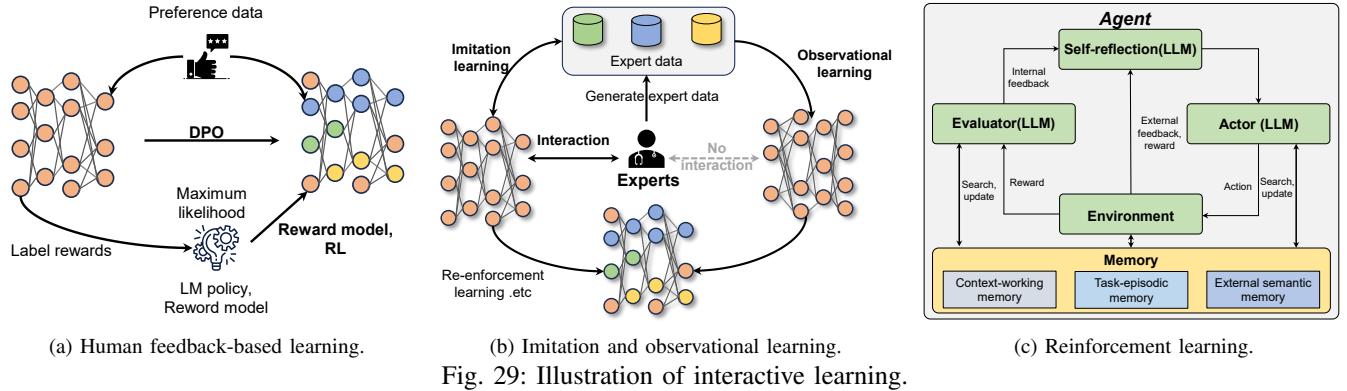


Fig. 29: Illustration of interactive learning.

occupancy-aligned distribution matching for improved LLM adaptation. Zhang *et al.* [264] propose a multimodal IL framework integrating visual–LiDAR fusion with reinforcement-guided prompt optimization. EMMA [55] aligns a VLM with an LLM expert via a DAgger-DPO algorithm to mitigate compounding errors. LLM-Personalize [265] bootstraps planning with demonstrations and iteratively refines policies through preference-aligned self-training.

3) *Observational Learning*: Unlike imitation learning, which directly maps expert state–action pairs, observational learning equips agents with behavioral competence by *interpreting observations* rather than replicating actions. This paradigm emphasizes building internal models of objectives, dynamics, and causal relations from multimodal inputs (videos, logs, text), enabling scalable self-supervised adaptation without explicit supervision. It is particularly effective for semantic grounding, long-horizon planning, and cross-modal alignment, where inference—not replication—drives behavior. Recent work illustrates diverse implementations: VELMA [266] verbalizes visual trajectories into textual prompts for in-context action prediction. ODA [267] applies an observe–act–reflect cycle with compact subgraphs for multi-hop reasoning and KG–FM synergy. CORY [268] coordinates pioneer and observer LLMs with shared rewards, enhancing robustness through co-adaptation.

4) *Reinforcement Learning*: In dynamic environments with long-horizon tasks and sparse rewards, static fine-tuning or fixed supervision often fail. Reinforcement learning (RL) offers an interaction-driven framework that optimizes policies via cumulative feedback, aligning actions with long-term outcomes without handcrafted rewards or expert demonstrations. This makes RL well-suited for FM-based agents in open-ended tasks such as semantic navigation, tool use, and multi-step instruction following (Fig. 29c). Recent work explores integrating RL with feedback and memory: Reflexion [93] enables verbal RL via self-reflective feedback and layered memory. REMEMBERER [269] augments Q-learning with episodic memory for experience reuse. Agent Q [270] couples MCTS with preference-guided off-policy optimization for web reasoning. FINCON [271] applies RL to financial decision-making through a Manager–Analyst multi-agent design. GLAM [272] grounds LLMs as policies in text environments via online PPO, improving efficiency and generalization.

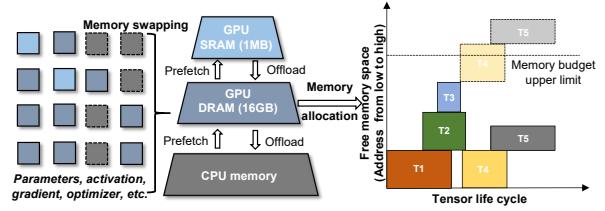


Fig. 30: Illustration of memory and parameter management.

E. Adaptive System Scheduling

Beyond algorithm-level optimization, adaptive system scheduling targets the performance–efficiency trade-off in agentic FM retraining (Fig. 19). It dynamically reallocates memory and compute within the Transformer graph by managing activation tensors and operator paths in real time. By exploiting FM retraining characteristics, it maximizes hardware utilization, enabling scalable, efficient, and on-the-fly test-time adaptation (Tab. XIV).

1) *Memory and Parameter Management*: Memory and parameter management complements algorithm-level PEFT (Sec. IV-B) by addressing system bottlenecks in FM adaptation. It emphasizes adaptive *allocation* and *swapping* of activations, gradients, optimizer states, and parameters to reduce fragmentation and peak usage, enabling scalable retraining under constrained hardware (Fig. 30).

a. Memory allocation. Static allocation fails under dynamic tensor shapes and irregular attention, leading to fragmentation. Recent work improves layout and reuse across layers and tiers: EDGE-LLM [58] adaptively places tensors via graph traversal over SRAM–DRAM–SSD; Memo [56] applies token-wise recomputation with bi-level planning; Silvestre *et al.* [57] exploit polyhedral dependence graphs for KV-cache reuse under dynamic shapes; and ZeRO-Infinity [273] shards states and offloads activations to CPU/NVMe for trillion-scale fine-tuning.

b. Memory swapping. Since memory-heavy components are not always active, swapping adaptively offloads them to CPU/SSD tiers and reloads on demand, balancing compute–memory trade-offs at the cost of I/O. Efficiency is enhanced by scheduling and compression: QLoRA [129] introduces paged optimizers with unified memory; Edge-LLM [58] searches cost models for tensor offloading; ProTrain overlaps swapping with compute; ES-MoE [276] pipelines expert-level caching; LSPOffload [275] combines sparse compres-

TABLE XIV: Summary of adaptive system scheduling for agentic FM retraining.

Categories		Technique highlight for improving	Year	Ref
Mobile LLM-adaptive system scheduling level (§IV-E)	Memory and parameter management (§IV-E1)	Model scheduling, offloading, hierarchical tensor placement.	2024	[58]
		Token-wise recomputation, bi-level memory planning.	2025	[56]
		Polyhedral dependence graphs, memory reuse, dynamic memory management.	2025	[57]
		Partition model states, offload to NVMe, memory-centric tiling.	2021	[273]
	Memory swapping (§IV-E1)	Sliding window eviction, tensor partitioning, in-place recomputation.	2023	[274]
		Dynamically allocate tensors, optimize offloading, hierarchical placement.	2024	[58]
		Paged Optimizers, automatic page migration, gradient checkpointing.	2023	[129]
		Sparse gradient compression, parallel cross-layer swapping.	2025	[275]
	Computation graph level (§IV-E2)	Offload expert parameters, VM-like prefetching, LRU caching.	2024	[276]
		Offload gradients and optimizer states, reduce bandwidth usage.	2024	[277]
		Recompute QK^T and softmax, minimize footprint, tile-wise backward pass.	2022	[59]
		Discard intermediate masked weights, retain input activations.	2025	[278]
	Activation recomputation (§IV-E2)	Fuse recomputation into single kernel.	2023	[60]
		Project updates into sparse subspaces, enable fine-tuning on GPUs, combine with checkpointing.	2025	[275]
		Exploit token-level sparsity, skip redundant activations.	2025	[279]
		Compile-time graph pruning, reorder scheduling, retain essential activations.	2023	[280]
	Operator fusion (§IV-E2)	Fuse matrix multiplication, reorder execution.	2022	[59]
		Replace split-K with split-Q, eliminate inter-warp communication.	2023	[60]
		Fuse low-rank update and mask, recompute on backward pass.	2025	[278]
		Identify commutative operators, merge operators, reuse context.	2024	[61]
	Operator reordering (§IV-E2)	Fuse LayerNorm and BatchMatMul.	2024	[280]
		Load K/V blocks, load Q sequentially, avoid full attention matrix.	2022	[59]
		Apply straight-through estimator, transform backward computation.	2025	[278]
		Polyhedral Dependence Graphs, optimize execution and memory.	2025	[57]
		Reorder gradient computation, early tensor release, reduce memory.	2023	[280]
		Asynchronous pipelining, break dependencies, improve throughput.	2024	[174]

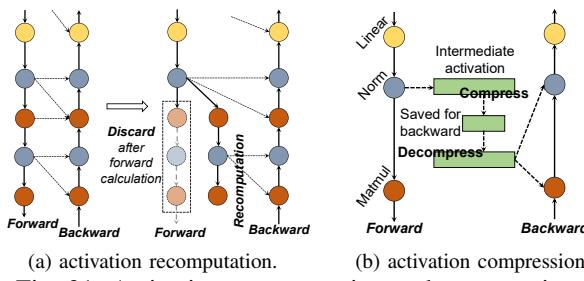


Fig. 31: Activation recomputation and compression.

sion with cross-layer bidirectional swapping; Elixir, PatrickStar [281], and Smart-Infinity [277] improve utilization via profiling, dynamic redistribution, and near-storage computing; while Zero-Offload jointly offloads data and compute across GPU–CPU–NVMe for scalable training.

2) *Computation Graph Optimization*: At the computation graph level, adaptive restructuring reduces FM adaptation overhead by pruning redundant states, refining execution order, and improving access patterns.

a. *Adaptive activation recomputation*. Recomputation lowers peak memory by discarding activations in forward and regenerating them in backward, trading compute for memory (Fig. 31a). FlashAttention-1/2/3 [59], [60], [174] progressively integrate fusion and asynchronous pipelining to reduce recomputation overhead, while LoRS [278] adaptively retains only input activations, recomputing masked weights to cut graph-tracking costs.

b. *Adaptive activation compression*. Compression directly shrinks activation storage via *low-rank projection*, *sparsity*, or *token skipping* (Fig. 31b). PockEngine [280] prunes and reorders computation graphs at compile time; LSP-Offload [275] projects gradients into sparse subspaces to complement checkpointing; LEMO [279] exploits token-level sparsity with fused operations, adaptively reducing memory to $1/N$ of baseline.

c. *Adaptive operator fusion*. Operator fusion dynamically merges tightly coupled ops (e.g., matmul, normalization, masking) into unified kernels, reducing memory traffic, shortening activation lifetimes, and alleviating I/O over-

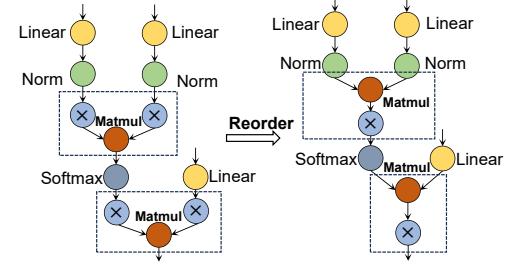


Fig. 32: Illustration of operator reordering.

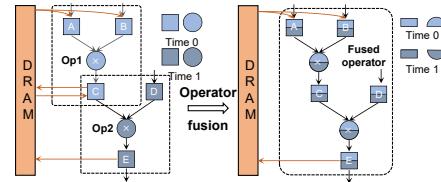


Fig. 33: Illustration of operator reordering.

head in resource-constrained fine-tuning (Fig. 32). FlashAttention [59], [60] fuses attention kernels to improve locality; PockEngine [280] compiles decomposed ops (e.g., LayerNorm+MatMul) into single kernels; Data-Juicer [61] applies context-aware, reordering-based fusion for pipeline efficiency; LoRS [278] fuses low-rank updates with masking into $\text{mask}_{\text{addmm}}$, adaptively discarding intermediates and recomputing during backprop.

d. *Adaptive operator reordering*. Operator reordering dynamically restructures execution order to shorten dependency chains, improve locality, and enable early release of intermediates under memory-constrained fine-tuning (Fig. 33). FlashAttention [59], [60], [174] exemplifies progressive reordering: preloading K, V to SRAM for on-chip aggregation (v1), deferring scaling with merged logsumexp (v2), and pipelined intra-/inter-warp scheduling (v3). PockEngine [280] compiles reordered gradient updates for in-place execution; LoRS [278] restructures backward passes via mask estimators and decomposed products; Silvestre *et al.* [57] use polyhedral dependence graphs to adaptively reorder across temporal/spatial dimen-

TABLE XV: Summary of distributed test-time FM retraining at agent networks.

Categories			Technique highlight for improving	Year	Ref
Distributed test-time adaptation for LLMs (§IV-F)	Federated fine-tuning (§IV-F1)	Communication efficiency (§IV-F1)	Select critical layer prompts, skip momentum exchange, optimize dual-side updates. Train adapters only, freeze backbone, transmit adapter configurations and parameters. Transmit lightweight PEFT modules, apply quantization and compression, unified communication. Transmit minimal trainable components, model communication cost.	2023 [63] 2023 [282] 2024 [283] 2024 [49]	
		Memory optimization (§IV-F1)	Restrict prompt updates to low-dimensional latent space, avoid backpropagation. Deploy low-rank adapters, load truncated submatrices, rank self-pruning. Select trainable weights, configure low-rank adapters.	2023 [284] 2024 [234] 2024 [285]	
		Computation efficiency (§IV-F1)	Freeze backbone, update lightweight adapters and heads, store pretrained weights locally. Activate shallow adapters, reconfigure adapter structures, cache cross-round activations. Optimize prompts via CMA-ES, update prompts with forward inference. Integrate resource-efficient operators, offload to multi-GPU and CPU, communication compression.	2023 [286] 2023 [282] 2023 [284] 2024 [283]	
		Data parallel adaptation (§IV-F2)	Train LoRA locally, synchronize updates via peer-to-peer. Drop transformer layers stochastically, tune PEFT modules, personalize layer sharing. Deferred initialization, sharding strategies, Communication overlap optimization.	2025 [287] 2025 [288] 2023 [289]	
	Distributed fine-tuning (§IV-F2)	Model parallel adaptation (§IV-F2)	Partition layers, retain trainable components, exchange activations and gradients. Partition layers, balance memory and throughput, synchronize intra-stage gradients. Split client/server submodels via weight importance, exchange activations and gradients. Hierarchical GPU-CPU workload, demand-priority scheduling.	2022 [290] 2024 [77] 2025 [291] 2024 [145]	
		Hybrid parallel adaptation (§IV-F2)	Adaptive pipelining reduces communication time and improves training throughput. Partition model layers, select cut layer, allocate server resources. Offload computation, retain lightweight adapters, decompose ranks adaptively. Delegate forward/backward computation, update adapters, mitigate device constraints.	2023 [292] 2025 [293] 2025 [291] 2025 [294]	

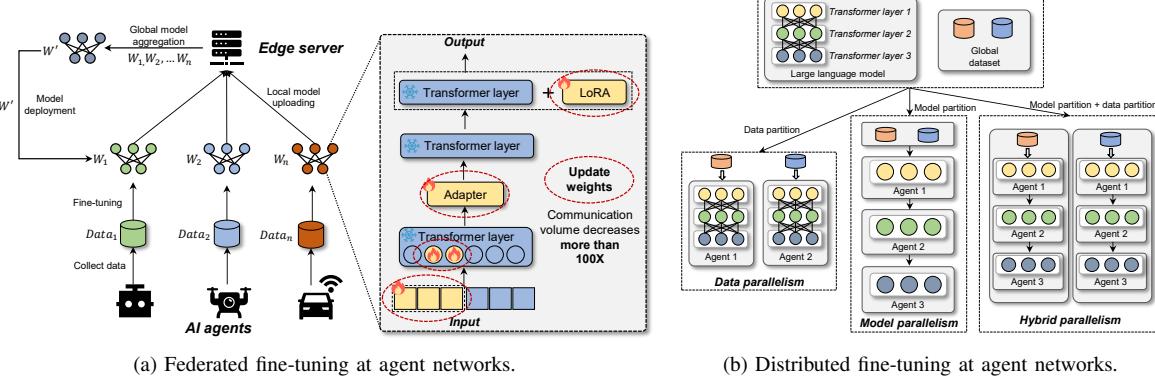


Fig. 34: Illustration of distributed test-time retraining of FMs at agent networks.

sions, improving scheduling efficiency in RLFT workflows.

F. Distributed Test-time FM Retraining at Agent Networks

Federated fine-tuning adapts FMs without centralizing data, but massive parameters, resource constraints, and heterogeneous distributions make full-model training infeasible. Thus, recent work focuses on lightweight, adaptive strategies:

1) *Federated Fine-tuning*: Federated fine-tuning adapts FMs without centralizing data, but massive parameters, resource constraints, and heterogeneous distributions make full-model training infeasible. Thus, recent work focuses on lightweight, adaptive strategies:

a. **Communication efficiency**. Transmit only small trainable modules (*e.g.*, adapters, prompts, biases), enhanced with quantization or compression. FedPepTAO [63] selects critical layers adaptively; AdaFL [282] freezes >99% of weights, cutting transmission by 126×; FS-LLM [283] integrates PEFT with quantized streaming for 1000× reduction; FedPEFT [49] formalizes adaptive cost-aware parameter exchange.

b. **Memory optimization**. Reduce local footprint by constraining updates to lightweight or quantized components. FedBPT [284] eliminates backprop with latent prompt tuning, lowering client memory 3×; HETLoRA [234] prunes heterogeneous low-rank modules, updating <5% parameters; FedPipe [285] combines selective LoRA ranks with 4/8-bit quantization, shrinking memory < 10% of baseline. Other

adaptive strategies include gradient-based selective layer tuning and adapter-based updates [286].

c. **Computation efficiency**. Computation efficiency in federated FM fine-tuning alleviates local training load by *limiting backpropagation*, *dynamically adapting trainable modules*, and *employing lightweight operators*. AdaFL [282] progressively activates shallow adapters with dynamic reconfiguration and activation caching to avoid redundant passes; FedBPT [284] eliminates gradients via forward-only black-box prompt tuning; FS-LLM [283] integrates mixed precision, gradient accumulation, and quantized communication, reusing frozen models across clients to reduce compute demand.

2) *Distributed FM Fine-tuning*: Distributed FM fine-tuning extends adaptation across collaborative device networks, enabling scalable training under resource-constrained and heterogeneous environments. By coordinating computation and storage in *peer-to-peer* or hierarchical *mobile-edge-cloud* systems, it overcomes single-device limits. Approaches fall into three categories: *data parallel*, *model parallel*, and *hybrid parallel adaptation*.

a. **Data parallel adaptation**. Replicates model states across devices and synchronizes updates, with *dynamic parameter-efficient modules* and dropout-based strategies reducing overhead. Dec-LoRA [287] trains LoRA locally and synchronizes updates peer-to-peer under non-IID data, DropPEFT [288] applies stochastic layer dropout with adaptive configuration to cut compute and communication on edge devices.

TABLE XVI: Summary of Dynamic Multi-modal FMs and Dynamic Cross-modal Alignment for Agentic AI Systems.

Categories		Technique highlight	Year	Ref
Dynamic multi-modal FMs (\$\S\$V-A)	Dynamic attention (\$\S\$V-A1)	Modality-specific attention, dynamic visual KV cache, periodic token update.	2025	[64]
		Grid sparse attention, adaptive stride, modality boundary permutation, pre-filling.	2025	[65]
		Selective token compression, softmax skipping, HilbertCurve permutation, quantized kernels.	2025	[70]
	Dynamic routing (\$\S\$V-A2)	Adaptive layer/head switching, latency scheduling.	2025	[295]
		Sparse MoE routing, entropy regularization, priority routing.	2022	[296]
		Adaptive deformable transformation, instruction-aware gating, sparse adapters.	2024	[66]
Dynamic multi-modal input adaptation (\$\S\$V-B)	Dynamic cross-modal alignment (\$\S\$V-B1)	Expert interaction, perturbation supervision, adaptive reweighting.	2025	[67]
		Sparse LoRA expert routing, domain conflict mitigation, auxiliary loss.	2024	[297]
		Trainable latent connections, adaptive block selection.	2025	[298]
		Linear adaptors for alignment, meta-response generation, language-level I/O alignment.	2024	[299]
		Cumulative model merging, realign modalities with replay-based connectors.	2025	[300]
		Bind diverse modalities through language, contrastive alignment, unified semantic space.	2025	[301]

b. Model parallel adaptation. Partitions FM layers/submodules across devices for pipeline-style computation, balancing memory and compute under edge constraints. PETALS [290] distributes Transformer layers while keeping PEFT modules local. Li *et al.* [293] apply split learning for LoRA, with CARD dynamically selecting cut layers to minimize latency/energy. Other adaptive schedulers include PipeMoE [292], which tunes pipeline degree and overlaps comm/compute, and APTMoE [145], which allocates workloads across GPUs/CPUs by expert popularity with demand-priority scheduling.

c. Hybrid parallel adaptation. Combines data, model, and offloading strategies to dynamically optimize memory, compute, and communication across heterogeneous tiers. PAC [77] adapts micro-batch scheduling with intra-stage AllReduce; HSPLITLoRA [291] partitions backbones server-side while tuning LoRA edge-side with adaptive rank decomposition; SplitLLM [294] extends to multi-tier edge–cloud, splitting forward/backward paths while updating adapters locally.

V. DYNAMIC MULTI-MODAL FMS IN AGENTIC AI SYSTEMS

Building on dynamic inference and test-time adaptation, agentic AI systems on mobile/edge platforms face heightened challenges in *multi-modal settings*. High-resolution vision, continuous speech, and heterogeneous sensor streams intensify redundancy and memory pressure, while complicating cross-modal alignment, consistency, and scalability. To address this, dynamic multi-modal FMs integrate *architectural- and input-level adaptations*, including: *dynamic attention*, *dynamic routing*, *adaptive cross-modal alignment*, and *token compression/pruning*. Together, these dynamic mechanisms balance efficiency and robustness for scalable multi-modal FM deployment under embedded resource constraints.

A. Dynamic Multi-modal FMs

Dynamic multi-modal FMs adapt their architectures to optimize computation, memory, and modality alignment in resource-constrained environments (Tab. XVI).

1) Dynamic Attention: Dynamic multi-modal FMs adapt architectures to optimize computation, memory, and modality alignment in resource-constrained environments (Tab. XVI). Core strategies include *dynamic attention* [64], [295], which generalizes sparse and hierarchical mechanisms to cross-modal settings by adaptively activating tokens, heads, or blocks according to modality-specific patterns and latency

budgets, thereby minimizing redundant computation and memory while preserving inference accuracy, and *dynamic routing*, which selectively activates modality experts to balance efficiency–accuracy trade-offs. For example, A-VL [64] reduces cost by dynamically selecting visual KV caches and exploiting the decay of text attention, MMInference [65] applies permutation-based sparse attention with modality-aware stride and phase search to cut pre-fill complexity, SpARGEAttention [70] combines token compression with sparse block prediction to skip redundant multiplications, and AdaLLaVA [295] employs a probabilistic scheduler to dynamically activate or skip Transformer blocks, heads, and neurons based on input content and resource budgets.

2) Dynamic Routing: Dynamic routing in multimodal FMs extends mixture-of-experts beyond unimodal efficiency optimization to address heterogeneous feature distributions, cross-modal alignment, and resource constraints. It dynamically activates modality- or interaction-specific experts to mitigate interference, handle missing modalities, and scale efficiently under limited compute and memory, typically through modality-aware gating, adaptive expert specialization, and sparse activation. *First*, PathWeave, FuseMoE, and LIMoE [296] employ *adaptive gating* to align heterogeneous features while avoiding efficiency collapse; *Second*, MoME [66], I²MoE [67], and CL-MoE introduce specialized or interaction-driven experts with *dynamic reweighting* to reduce interference and enhance continual learning; *Third*, MoTE, EvoMoE [302], DeepSeek-VL2, LLaVA-MoLE [297], and Uni-MoE adopt token-level or evolving sparse routing with *adaptive load balancing* to optimize scalability and efficiency in resource-constrained deployments.

B. Dynamic Multi-modal Input Adaptation

Dynamic multi-modal input adaptation addresses the core challenge of processing heterogeneous and redundant inputs under resource constraints in embedded and edge-deployed agentic AI systems. It must handle high-resolution vision, continuous speech, and dense sensor streams, which impose heavy demands on memory, computation, and energy. Efficient adaptation therefore requires input-aware strategies that dynamically allocate resources and sustain cross-modal alignment during real-time inference and retraining.

1) Dynamic Cross-modal Alignment: Dynamic cross-modal alignment [298]–[300] enables multimodal FMs on edge and mobile devices to adaptively include or exclude modalities based on system availability, while supporting intra-modality

TABLE XVII: Summary of multi-modal token compression for dynamic multi-modal FMs in agentic AI systems.

Categories		Technique highlight	Year	Ref
Multi-modal token compression (\$V\$-\$B\$2)	Token pruning (\$V\$-\$B\$2)	Prune low-attention tokens, bypass deep layers.	2024	[303]
		Select salient tokens, prune redundancies, merge keys.	2024	[304]
		Optimal transport for pruning, Sinkhorn estimation, prefilling pruning.	2025	[305]
	Token merging (\$V\$-\$B\$2)	Fuse visual tokens, query-based merging.	2025	[306]
		Adaptive pooling for vision token compression, semantic abstraction.	2024	[307]
		Similarity-driven token merging, length reduction, threshold control.	2023	[308]
	Adaptive sampling (\$V\$-\$B\$2)	Vision-language matching, recursive partitioning.	2025	[309]
		AnyRes encoding, bilinear interpolation, adaptive resampling.	2024	[310]
		Block-wise streaming, TMRoPE, dynamic resampling.	2025	[311]

domain adaptation to ensure robustness under heterogeneous and resource-constrained sensing. Approaches span three directions: *First*, for *new modality adaptation*, MPnP [298] introduces nonlinear key-value aligners with latent connections to dynamically control injection depth, and ModaVerse [299] aligns language I/O via single-stage tuning to reduce projection-instruction mismatch; *Second*, for *incremental adaptation and forgetting mitigation*, MERA [300] combines cumulative average merging with selective replay, MoInCL [312] leverages instruction-guided pseudo-targets, BABEL applies sequential binary alignments with gradient-norm weighting, and SEMI [301] generates LoRA adapters via a hypernetwork to enhance cross-modal consistency; *Third*, for *dynamic fusion*, PathWeave adopts an adapter-in-adapter MoE framework for adaptive modality coordination, while FuseMoE employs instance-level gating to dynamically fuse features, mitigating redundancy and semantic drift.

2) *Multi-modal Token Compression*: Multi-modal token compression addresses the quadratic cost of self-attention in handling long sequences from high-resolution vision, continuous speech, and heterogeneous sensors, which is prohibitive for mobile and edge deployment. Recent approaches emphasize dynamic and adaptive compression by pruning, merging, or sampling tokens to jointly balance efficiency and accuracy. *First*, *token pruning* methods such as FastV [303], VTV [313], and DivPrune dynamically discard redundant or low-importance tokens based on attention statistics, divergence, or diversity maximization, while TopV [305] applies transport-based selection for latency reduction. *Second*, *token merging* techniques including A-ToMe [308], LOOK-M, and LLaVA-Mini [306] compact semantically similar tokens across modalities into fewer representations, reducing sequence length while preserving contextual coherence. *Third*, *adaptive sampling* strategies such as AKS [309], LongVU [314], and Qwen2.5-VL [311] further enhance efficiency by dynamically selecting task-relevant video frames, image patches, or spatiotemporal regions under varying resource budgets, while DeepSeek-VL2 combines global-local tiling for fine-grained yet efficient visual understanding.

VI. AGENTIC AI APPLICATIONS

Agentic AI systems are rapidly extending into diverse domains, imposing various adaptation and efficiency challenges.

A. Representative Applications

We first present four representative applications, illustrating the diversity of interaction mechanisms, from physical manipulation to digital automation and creative generation, highlighting adaptive and resource-efficient demands.

Embody Agents. Embodied agents, including robots [11], [21], drones [315], grippers [316], and wearables [317], perceive and act in physical environments under strict latency, energy, and compute constraints. Adaptation is critical for multimodal perception, embodied interaction, and sim-to-real transfer. PaLM-SSE [71] integrates continuous sensor modalities into an LLM for robotic planning and manipulation, HuggingGPT [72] orchestrates perception and action through specialized models, and ReCA [318] co-designs algorithms, systems, and hardware for real-time multi-agent collaboration.

GUI Agents. GUI agents perceive, reason about, and act upon graphical user interfaces, combining multimodal inputs (screenshots, text, speech, contextual cues) with actions via clicks, taps, or API calls. Their applications span workflow automation [319], accessibility [320], web/mobile navigation [321], and human-computer interaction [322]. Adaptive fusion and reasoning are central to recent advances: MP-GUI [73] employs modality-specific perceivers with adaptive fusion for robust GUI understanding; GUI-World [74] benchmarks dynamic GUI interactions and shows persistent challenges for LLMs; InfiGUI-R1 [323] transitions from reactive to reasoning-based agents with spatial distillation and RL for error recovery; and Mirage-1 [324] develops hierarchical multimodal skills with adaptive Monte Carlo Tree Search for long-horizon GUI tasks.

Generative Agents. Generative agents produce new content across modalities, *e.g.*, text, image, audio, code, and media, while operating under latency and resource constraints. They power diverse applications including summarization [325], dialogue [326], translation [327], creative tools [328], code generation [329], and multimodal synthesis [330]. Beyond content creation, generative agents also model human behavior. Generative simulations of 1,000 People [75] replicate personality traits and survey responses with 85% accuracy, while AgentSociety [331] scales to 10,000 agents and 5M+ interactions to study social phenomena like polarization. Distributed deployments [76] emphasize modularity, energy efficiency, and privacy in heterogeneous edge environments. To sustain output quality (coherence, factuality, creativity) under resource limits, these agents must dynamically adjust complexity, caching, or approximations (*e.g.*, quantization, lightweight modules). Yet continual adaptation in generative settings, especially for repeated user/environment interactions, remains underexplored.

Personal Assistive Agents. Personal assistive agents operate on mobile/edge devices to support health monitoring [332], accessibility [333], and productivity [334]. Unlike cloud-based assistants, they rely on local data and dynamically adapt to user-specific contexts, shifting environments, and resource variability under tight privacy and power constraints. Advances

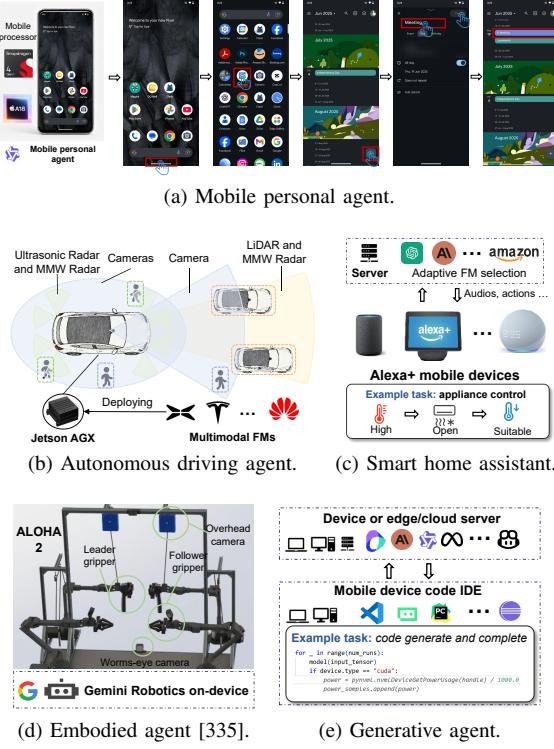


Fig. 35: Illustration of case study implementation.

such as Pluto/Charon (PAC) [77] enable lightweight on-device personalization, Privacyasst [78] sanitize prompts to mitigate data leakage. They highlight the dual challenge of achieving adaptive personalization with scarce training data and sustaining resource-aware operation via quantization, pruning, and elastic inference.

B. Case Study Implementation

To illustrate how agentic AI operates in practice, we show representative deployments across mobile, automotive, smart home, robotics, and software development (Fig. 35).

Mobile Personal Agent. Alibaba’s GUI-Owl family exemplifies a cross-device personal assistant. The lightweight GUI-Owl-7B [15] runs locally on smartphones and PCs for shopping or scheduling, while the larger GUI-Owl-32B is deployed on home-edge hubs for complex reasoning. Built on Mobile-Agent-v3, it integrates *manager*, *worker*, *reflector*, and *notetaker* agents, enabling adaptive task coordination, real-time correction, and cross-device memory.

Autonomous Driving Agent. XPeng Motors [336] compresses a 72B-parameter World FM via RL and distillation to run on in-vehicle NVIDIA Orin-X units. With XNet for perception, XPlanner for trajectory prediction, and XBrain for reasoning, the system dynamically balances accuracy-latency trade-offs to support safety-critical functions such as adaptive braking and lane switching.

Smart Home Voice Assistant. Amazon’s Alexa+ [337] combines on-device wake-word detection with edge-hosted Nova FM and external LLMs (*e.g.*, Claude). This hybrid design dynamically allocates inference across devices, ensuring low-latency responses while scaling to complex orchestration tasks like API-based appliance control and personalized dialogue.

Embodied Agent. DeepMind’s Gemini Robotics [338] runs multimodal reasoning on edge servers while deploying lightweight decoders on robots (*e.g.*, ALOHA 2 [335]). By adaptively fusing 3D perception, planning, and safety checks with low-latency motor control, it enables real-world dexterous manipulation from assembly to healthcare delivery.

Generative Agent. ByteDance’s Trae integrates Doubao-1.5-Pro [339] and DeepSeek-R1 [106] into IDE-native workflows. Through SOLO and Chat modes, plus MCP-based automation, it adapts code generation, debugging, and pull request analysis to developer needs. The 6A workflow (Align–Architect–Atomize–Approve–Automate–Assess) improves individual productivity by 40%, demonstrating adaptive generative agents for enterprise-scale collaboration.

C. Models and Datasets

Recent FMs have been increasingly explored for agentic deployment, with some explicitly designed for mobile/edge efficiency and others requiring post-hoc compression and dynamic adaptation. Representative efforts include LLaMA [26], OPT [340], and SAM [341] from *Meta*; PaLM [108], Gemini [342], and T5 [343] from *Google*; Phi [344], Orca [345], and Kosmos [346] from *Microsoft*; Doubao [339] from *ByteDance*; and GPT [29], [347] and DALL-E [348] from *OpenAI*. While FMs benefit from massive pretraining corpora (*e.g.*, GPT-4 with 13T tokens), domain-specific agentic applications remain constrained by fragmented data ecosystems, demanding adaptive and resource-efficient retraining strategies.

Evaluation datasets span multiple modalities. For *language*, BIG-Bench [349] and HELM [350] benchmark reasoning, planning, and dialogue. For *vision*, Open X-Embodiment [351] and ScanNet++ [352] support robotic perception and manipulation. For *multimodality*, ScienceQA [353], MMBench [354], and SEED-Bench [355] test cross-modal reasoning and grounded interaction. For *embodied tasks*, BEHAVIOR-1K [356], MineDojo [357], and CALVIN [358] capture long-horizon perception-action loops.

Despite this progress, current benchmarks insufficiently reflect the dynamic, resource-constrained conditions of mobile/edge deployment. Developing adaptive, domain-specific models and datasets that capture environmental variability, multimodal asynchrony, and real-time feedback remains a critical challenge for evaluating agentic AI in practice.

D. Inference Engines

To support the performance–portability demands of agentic AI, inference engines integrate resource-efficient optimizations to adapt FMs across heterogeneous hardware, from edge servers to highly constrained mobile/embedded devices [163], [359]–[364]. *llama.cpp* [359] demonstrates on-device deployment via quantization, SIMD kernels, heterogeneous backends, and memory mapping, while *vLLM* [163] targets edge/server environments with PagedAttention for KV-cache management and continuous batching, trading portability for large-scale throughput. Cross-platform frameworks such as TensorRT-LLM and OpenVINO [365] accelerate inference on GPUs/CPUs, whereas Core ML [360] and TensorFlow

Lite [361] specialize in mobile/wearables with quantization, pruning, and hardware-specific kernels. Lightweight engines like MNN [362], NCNN [363], and MindSpore Lite [364] focus on IoT and Android devices.

VII. OPEN ISSUES

This section outlines key challenges and potential directions.

A. Elastic Inference for Perception–Action Loop

A central challenge for agentic AI lies in reconciling static foundation models (FMs) with the dynamic perception–action loops of real-world environments. Current FMs excel at symbol processing over text, vision, or multimodal data, yet lack elastic inference that adapts to real-time sensing, long-horizon control, and fluctuating hardware resources, leaving them misaligned with latency, efficiency, and adaptability demands in embedded or embodied deployments. Recent explorations, such as VLA models for end-to-end perception–action coupling [336], [338], hierarchical reasoning for modular control, and world models for causal dynamics [366], show promise but remain immature, failing to close the gap between massive FMs and asynchronous, resource-constrained settings. Key *gaps* include the absence of elastic architectural adaptation, where perception, reasoning, memory, and action submodules scale independently across heterogeneous devices, and the lack of joint resource–model co-optimization that aligns hardware scheduling with submodule reconfiguration. Future solutions may involve meta-controllers that dynamically assemble task- and domain-specific FMs, and modular operators that support composable, elastic scaling and distributed offloading across agents and edge nodes.

B. Lightweight yet Generalizable Physical Intelligence

Despite advances in text and multimodal reasoning, current FMs remain misaligned with the requirements of physical intelligence, where agents must sustain closed-loop perception–planning–control under real-world uncertainty [29], [339]. Lightweight models lack the capacity to bridge abstract reasoning with fine-grained action, handle noisy multimodal feedback, or maintain long-horizon causal dependencies, while large embodied FMs (*e.g.*, 40~140B multimodal models) still suffer from scarce robotic data and persistent sim-to-real gaps. Pretrain–finetune pipelines further struggle in sequential, causal operations, limiting generalization. The central challenge is delivering models that are both lightweight and generalizable: capable of fusing heterogeneous sensor streams, embedding contextual memory for adaptive planning, and producing executable action sequences under strict latency, memory, and energy budgets. Progress will require advances in multimodal fusion, domain generalization, and resource-aware adaptation, making lightweight yet generalizable physical intelligence a key frontier for agentic AI.

C. Responsive Online FM Adaptation

Agentic AI systems on autonomous platforms (*e.g.*, drones, underwater robots, quadrupeds) often lack stable cloud connectivity, requiring foundation models to adapt online under

strict memory, compute, and energy limits. The challenge is sustaining performance when full retraining is infeasible, data streams are unlabeled, and source data is inaccessible, creating risks of error propagation and catastrophic forgetting. Balancing accuracy and efficiency hinges on selecting which data to use and which modules or neurons to update. Promising directions include PEFT, prompt learning, interactive learning, and memory-augmented adaptation to shrink trainable parameters, combined with memory management, execution scheduling, and distributed retraining to align updates with hardware constraints. Yet integrating these techniques with label-free streaming, limited backpropagation, and missing source data remains unresolved. Moreover, current autoregressive architectures, rooted in probabilistic generation, are brittle for long-horizon reasoning, compounding errors over time. Addressing these gaps calls for uncertainty-aware data selection, real-time relearning, and joint algorithm–system co-design that couples adaptive parameter updates with device-aware scheduling.

D. Real-time Distributed Multi-modal Sensing

Mobile and edge agents with heterogeneous sensors (*e.g.*, cameras, LiDAR, RF) enable multimodal FMs for autonomous driving and smart environments, yet real-time sensing is fundamentally constrained by *asynchrony*, *redundancy*, and *communication overhead*. Sensor streams differ in rate and latency (*e.g.*, camera *vs.* LiDAR under a 100 Mbps link show a 1:4 imbalance, introducing 40 ms delays). Waiting for slow modalities increases latency, while discarding them reduces accuracy, creating an inherent latency–accuracy–communication trade-off that current synchronous, full-modality MLLMs cannot resolve [311], [367]. Emerging paradigms such as predict–verify [368] and early-exit [137], [138], [146] partly mitigate cost, but lack principled ways to capture *modality affinity*, a unified measure of cross-modal consistency, complementarity, and communication relevance across distributed agents and dynamic inputs. Key challenges remain: (*i*) building generalizable models to evaluate and exploit modality affinity under asynchrony and bandwidth limits, (*ii*) designing *non-blocking imputation* and communication-efficient fusion for missing or delayed modalities, and (*iii*) enabling adaptive predict–verify loops with rollback to balance timeliness, accuracy, and communication cost. Addressing these challenges demands holistic co-design of *FM architectures*, *runtime scheduling*, and *adaptive network protocols* to jointly optimize sensing, computation, and communication for robust real-world deployment.

E. Efficient Multi-agent Collaboration

As FMs grow and edge devices remain resource-limited, multi-agent collaboration is increasingly necessary to overcome single-agent bottlenecks. Yet most existing methods rely on static compile-time partitioning, where model splitting dictates task dispatch. This rigid coupling forces frequent recompilation under dynamic conditions, incurring high latency and overhead. The problem is worsened by Transformer-based FMs, whose intermediate features often exceed raw

input size [11], creating prohibitive transmission costs that hinder distributed deployment. The core challenge is enabling *elastic, runtime collaboration* that dynamically balances model partitioning and communication. This requires *i)* rapid repartitioning and adaptive scheduling across heterogeneous devices, *ii)* reducing transmission overhead from large intermediate activations, and *iii)* communication-efficient protocols that sustain responsiveness under bandwidth, latency, and energy limits. While recent advances in swarm robotics, federated perception [369], and structured coordination frameworks [370] highlight potential, achieving scalable, communication-aware, and resource-efficient collaboration in dynamic environments remains open issues.

F. Interactive and Collaborative Human–AI Systems

Human–AI collaboration requires agents that can perceive intent, interpret context, and generate adaptive responses through natural, multimodal interaction. The central challenge lies in sustaining *real-time, resource-aware adaptation* on mobile and edge platforms while integrating continuous feedback from users and environments. Current FMs, which rely mainly on parameter updates, remain limited for dynamic interaction. Open problems include *i)* enabling cognition updates through memory, external knowledge, and prompt-driven control; *ii)* unifying perception–language–action pipelines under resource constraints; and *iii)* balancing responsiveness, personalization, and energy efficiency. Humans must be treated not only as data annotators but also as interactive partners, providing feedback during inference to refine reasoning, guide actions, and establish trust in collaborative systems.

VIII. CONCLUSION

Agentic AI marks a paradigm shift, embedding foundation models into mobile, embedded, and edge systems where adaptivity and resource efficiency are necessities rather than optimizations. Unlike cloud settings, real-world deployments face diverse tasks, dynamic environments, and constrained hardware, demanding elastic inference and online adaptation. This survey clarifies key concepts, presents a taxonomy of enabling techniques, and highlights applications spanning embodied, GUI, generative, and personal agents. We also emphasize distributed coordination, where communication and scheduling shape multi-agent efficiency. Looking forward, progress will depend on unified benchmarks, algorithm–system–hardware co-design, trustworthy adaptation pipelines, and robust low-latency communication. We aim to motivate future work toward scalable, reliable, and personalized agentic AI bridging powerful FMs with resource-limited real-world environments.

ACKNOWLEDGMENTS

This work was partially supported by the National Science Fund for Distinguished Young Scholars (No. 62025205) and the National Natural Science Foundation of China (No. 62522215, 62532009, 6247074224).

REFERENCES

- [1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.
- [2] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, “An image is worth 16x16 words: Transformers for image recognition at scale,” *arXiv preprint arXiv:2010.11929*, 2020.
- [3] S. Russell, P. Norvig, and A. Intelligence, “A modern approach,” *Artificial Intelligence. Prentice-Hall, Englewood Cliffs*, vol. 25, no. 27, pp. 79–80, 1995.
- [4] E. R. Teoh and D. G. Kidd, “Rage against the machine? google’s self-driving cars versus human drivers,” *Journal of safety research*, vol. 63, pp. 57–60, 2017.
- [5] Y. Hu, Q. Xie, V. Jain, J. Francis, J. Patrikar, N. Keetha, S. Kim, Y. Xie, T. Zhang, H.-S. Fang *et al.*, “Toward general-purpose robots via foundation models: A survey and meta-analysis,” *arXiv preprint arXiv:2312.08782*, 2023.
- [6] T. Gunter, Z. Wang, C. Wang, R. Pang, A. Narayanan, A. Zhang, B. Zhang, C. Chen, C.-C. Chiu, D. Qiu *et al.*, “Apple intelligence foundation language models,” *arXiv preprint arXiv:2407.21075*, 2024.
- [7] P. Delgado-Santos, G. Stragapede, R. Tolosana, R. Guest, F. Deravi, and R. Vera-Rodriguez, “A survey of privacy vulnerabilities of mobile device sensors,” *ACM Computing Surveys (CSUR)*, vol. 54, no. 11s, pp. 1–30, 2022.
- [8] P. Dhilleswararao, S. Boppu, M. S. Manikandan, and L. R. Cenkeramaddi, “Efficient hardware architectures for accelerating deep neural networks: Survey,” *IEEE access*, vol. 10, pp. 131788–131828, 2022.
- [9] P. Mach and Z. Becvar, “Mobile edge computing: A survey on architecture and computation offloading,” *IEEE communications surveys & tutorials*, vol. 19, no. 3, pp. 1628–1656, 2017.
- [10] R. Singh and S. S. Gill, “Edge ai: a survey,” *Internet of Things and Cyber-Physical Systems*, vol. 3, pp. 71–92, 2023.
- [11] M. Xu, W. Yin, D. Cai, R. Yi, D. Xu, Q. Wang, B. Wu, Y. Zhao, C. Yang, S. Wang *et al.*, “A survey of resource-efficient llm and multimodal foundation models,” *arXiv preprint arXiv:2401.08092*, 2024.
- [12] Z. Zhou, X. Ning, K. Hong, T. Fu, J. Xu, S. Li, Y. Lou, L. Wang, Z. Yuan, X. Li *et al.*, “A survey on efficient inference for large language models,” *arXiv preprint arXiv:2404.14294*, 2024.
- [13] W. Fedus, B. Zoph, and N. Shazeer, “Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity,” *Journal of Machine Learning Research*, vol. 23, no. 120, pp. 1–39, 2022.
- [14] N. Du, Y. Huang, A. M. Dai, S. Tong, D. Lepikhin, Y. Xu, M. Krikun, Y. Zhou, A. W. Yu, O. Firat *et al.*, “Glam: Efficient scaling of language models with mixture-of-experts,” in *International conference on machine learning*. PMLR, 2022, pp. 5547–5569.
- [15] J. Ye, X. Zhang, H. Xu, H. Liu, J. Wang, Z. Zhu, Z. Zheng, F. Gao, J. Cao, Z. Lu *et al.*, “Mobile-agent-v3: Foundamental agents for gui automation,” *arXiv preprint arXiv:2508.15144*, 2025.
- [16] S. Harrer, R. V. Rane, and R. E. Speight, “Generative ai agents are transforming biology research: high resolution functional genome annotation for multiscale understanding of life,” *EBioMedicine*, vol. 109, 2024.
- [17] L. Yee, M. Chui, R. Roberts, and S. Xu, “Why agents are the next frontier of generative ai,” *McKinsey Digital Practice*, 2024.
- [18] D. Aladin, O. Varlamov, D. Chuvikov, V. Chernenkiy, E. Smelkova, and A. Baldin, “Logic-based artificial intelligence in systems for monitoring the enforcing traffic regulations,” in *IOP Conference Series: Materials Science and Engineering*, vol. 534, no. 1. IOP Publishing, 2019, p. 012025.
- [19] R. Calegari, G. Ciatto, V. Mascardi, and A. Omicini, “Logic-based technologies for multi-agent systems: a systematic literature review,” *Autonomous Agents and Multi-Agent Systems*, vol. 35, no. 1, p. 1, 2021.
- [20] J. Xie, Z. Chen, R. Zhang, X. Wan, and G. Li, “Large multimodal agents: A survey,” *arXiv preprint arXiv:2402.15116*, 2024.
- [21] J. Duan, S. Yu, H. L. Tan, H. Zhu, and C. Tan, “A survey of embodied ai: From simulators to research tasks,” *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 6, no. 2, pp. 230–244, 2022.
- [22] Y. Wu, P. Zhang, M. Gu, J. Zheng, and X. Bai, “Embodied navigation with multi-modal information: A survey from tasks to methodology,” *Information Fusion*, vol. 112, p. 102532, 2024.
- [23] A. M. Turing, “Computing machinery and intelligence,” in *Parsing the Turing test: Philosophical and methodological issues in the quest for the thinking computer*. Springer, 2007, pp. 23–65.

- [24] S. Liu, B. Guo, C. Fang, Z. Wang, S. Luo, Z. Zhou, and Z. Yu, “Enabling resource-efficient aiot system with cross-level optimization: A survey,” *IEEE Communications Surveys & Tutorials*, vol. 26, no. 1, pp. 389–427, 2023.
- [25] H.-I. Liu, M. Galindo, H. Xie, L.-K. Wong, H.-H. Shuai, Y.-H. Li, and W.-H. Cheng, “Lightweight deep learning for resource-constrained environments: A survey,” *ACM Computing Surveys*, vol. 56, no. 10, pp. 1–42, 2024.
- [26] H. Touvron, T. Lavigra, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample, “Llama: Open and efficient foundation language models,” *arXiv preprint arXiv:2302.13971*, 2023.
- [27] J. Wei, Y. Tay, R. Bommasani, C. Raffel, B. Zoph, S. Borgeaud, D. Yogatama, M. Bosma, D. Zhou, D. Metzler *et al.*, “Emergent abilities of large language models,” *arXiv preprint arXiv:2206.07682*, 2022.
- [28] R. Schaeffer, B. Miranda, and S. Koyejo, “Are emergent abilities of large language models a mirage?” *Advances in neural information processing systems*, vol. 36, pp. 55 565–55 581, 2023.
- [29] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, “Language models are few-shot learners,” *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.
- [30] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou *et al.*, “Chain-of-thought prompting elicits reasoning in large language models,” *Advances in neural information processing systems*, vol. 35, pp. 24 824–24 837, 2022.
- [31] J. S. Park, J. O’Brien, C. J. Cai, M. R. Morris, P. Liang, and M. S. Bernstein, “Generative agents: Interactive simulacra of human behavior,” in *Proceedings of the 36th annual ACM symposium on user interface software and technology*, 2023, pp. 1–22.
- [32] K. Udagawa, Y. Saito, and H. Saruwatari, “Human-in-the-loop speaker adaptation for dnn-based multi-speaker tts,” *arXiv preprint arXiv:2206.10256*, 2022.
- [33] W. Xiang, H. Zhu, S. Lou, X. Chen, Z. Pan, Y. Jin, S. Chen, and L. Sun, “Simuser: Generating usability feedback by simulating various users interacting with mobile applications,” in *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*, 2024, pp. 1–17.
- [34] A. Pritzel, B. Urias, S. Srinivasan, A. P. Badia, O. Vinyals, D. Hassabis, D. Wierstra, and C. Blundell, “Neural episodic control,” in *International conference on machine learning*. PMLR, 2017, pp. 2827–2836.
- [35] W. Zhou, Y. E. Jiang, R. Cotterell, and M. Sachan, “Efficient prompting via dynamic in-context learning,” *arXiv preprint arXiv:2305.11170*, 2023.
- [36] F. Xu, W. Shi, and E. Choi, “Recomp: Improving retrieval-augmented lms with compression and selective augmentation,” *arXiv preprint arXiv:2310.04408*, 2023.
- [37] A. Mitra, L. Del Corro, G. Zheng, S. Mahajan, D. Rouhana, A. Codas, Y. Lu, W.-g. Chen, O. Vrousgos, C. Rosset *et al.*, “Agentin-struct: Toward generative teaching with agentic flows,” *arXiv preprint arXiv:2407.03502*, 2024.
- [38] X. Ning, Z. Lin, Z. Zhou, Z. Wang, H. Yang, and Y. Wang, “Skeleton-of-thought: Prompting llms for efficient parallel generation,” *arXiv preprint arXiv:2307.15337*, 2023.
- [39] M. Nye, A. J. Andreassen, G. Gur-Ari, H. Michalewski, J. Austin, D. Bieber, D. Dohan, A. Lewkowycz, M. Bosma, D. Luan *et al.*, “Show your work: Scratchpads for intermediate computation with language models,” 2021.
- [40] W. Hua, Z. Dai, H. Liu, and Q. Le, “Transformer quality in linear time,” in *International conference on machine learning*. PMLR, 2022, pp. 9099–9117.
- [41] E. Frantar and D. Alistarh, “Sparsegpt: Massive language models can be accurately pruned in one-shot,” 2023. [Online]. Available: <https://arxiv.org/abs/2301.00774>
- [42] A. Liu, B. Feng, B. Xue, B. Wang, B. Wu, C. Lu, C. Zhao, C. Deng, C. Zhang, C. Ruan *et al.*, “Deepseek-v3 technical report,” *arXiv preprint arXiv:2412.19437*, 2024.
- [43] Z. Zeng, Y. Hong, H. Dai, H. Zhuang, and C. Chen, “Consistentee: A consistent and hardness-guided early exiting method for accelerating language models inference,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 17, 2024, pp. 19 506–19 514.
- [44] Z. Zhang, Y. Sheng, T. Zhou, T. Chen, L. Zheng, R. Cai, Z. Song, Y. Tian, C. Ré, C. Barrett *et al.*, “H2o: Heavy-hitter oracle for efficient generative inference of large language models,” *Advances in Neural Information Processing Systems*, vol. 36, pp. 34 661–34 710, 2023.
- [45] Z. Liu, A. Desai, F. Liao, W. Wang, V. Xie, Z. Xu, A. Kyriolidis, and A. Shrivastava, “Scissorhands: Exploiting the persistence of importance hypothesis for llm kv cache compression at test time,” *Advances in Neural Information Processing Systems*, vol. 36, pp. 52 342–52 364, 2023.
- [46] T. Shin, Y. Razeghi, R. L. Logan IV, E. Wallace, and S. Singh, “Autoprompt: Eliciting knowledge from language models with automatically generated prompts,” *arXiv preprint arXiv:2010.15980*, 2020.
- [47] S. Lee, J. Choi, J. Lee, M. H. Wasi, H. Choi, S. Ko, S. Oh, and I. Shin, “Mobilegpt: Augmenting llm with human-like app memory for mobile task automation,” in *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking*, 2024, pp. 1119–1133.
- [48] T. Guo, S. Guo, J. Wang, X. Tang, and W. Xu, “Promptfl: Let federated participants cooperatively learn prompts instead of models–federated learning in age of foundation model,” *IEEE Transactions on Mobile Computing*, vol. 23, no. 5, pp. 5179–5194, 2023.
- [49] G. Sun, U. Khalid, M. Mendieta, P. Wang, and C. Chen, “Exploring parameter-efficient fine-tuning to enable foundation models in federated learning,” in *2024 IEEE International Conference on Big Data (BigData)*. IEEE, 2024, pp. 8015–8024.
- [50] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, W. Chen *et al.*, “Lora: Low-rank adaptation of large language models.” *ICLR*, vol. 1, no. 2, p. 3, 2022.
- [51] J. Mu, X. Li, and N. Goodman, “Learning to compress prompts with gist tokens,” *Advances in Neural Information Processing Systems*, vol. 36, pp. 19 327–19 352, 2023.
- [52] X. Wang, C. Li, Z. Wang, F. Bai, H. Luo, J. Zhang, N. Jovic, E. P. Xing, and Z. Hu, “Promptagent: Strategic planning with language models enables expert-level prompt optimization,” *arXiv preprint arXiv:2310.16427*, 2023.
- [53] N. Ratner, Y. Levine, Y. Belinkov, O. Ram, I. Magar, O. Abend, E. Karpas, A. Shashua, K. Leyton-Brown, and Y. Shoham, “Parallel context windows for large language models,” *arXiv preprint arXiv:2212.10947*, 2022.
- [54] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray *et al.*, “Training language models to follow instructions with human feedback,” *Advances in neural information processing systems*, vol. 35, pp. 27 730–27 744, 2022.
- [55] Y. Yang, T. Zhou, K. Li, D. Tao, L. Li, L. Shen, X. He, J. Jiang, and Y. Shi, “Embody multi-modal agent trained by an llm from a parallel textworld,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2024, pp. 26 275–26 285.
- [56] P. Zhao, H. Zhang, F. Fu, X. Nie, Q. Liu, F. Yang, Y. Peng, D. Jiao, S. Li, J. Xue *et al.*, “Memo: Fine-grained tensor management for ultra-long context llm training,” *Proceedings of the ACM on Management of Data*, vol. 3, no. 1, pp. 1–28, 2025.
- [57] P. F. Silvestre and P. Pietzuch, “Systems opportunities for llm fine-tuning using reinforcement learning,” in *Proceedings of the 5th Workshop on Machine Learning and Systems*, 2025, pp. 90–99.
- [58] Z. Yu, Z. Wang, Y. Li, R. Gao, X. Zhou, S. R. Bommu, Y. Zhao, and Y. Lin, “Edge-llm: Enabling efficient large language model adaptation on edge devices via unified compression and adaptive layer voting,” in *Proceedings of the 61st ACM/IEEE Design Automation Conference*, 2024, pp. 1–6.
- [59] T. Dao, D. Fu, S. Ermon, A. Rudra, and C. Ré, “Flashattention: Fast and memory-efficient exact attention with io-awareness,” *Advances in neural information processing systems*, vol. 35, pp. 16 344–16 359, 2022.
- [60] T. Dao, “Flashattention-2: Faster attention with better parallelism and work partitioning,” *arXiv preprint arXiv:2307.08691*, 2023.
- [61] D. Chen, Y. Huang, Z. Ma, H. Chen, X. Pan, C. Ge, D. Gao, Y. Xie, Z. Liu, J. Gao *et al.*, “Data-juicer: A one-stop data processing system for large language models,” in *Companion of the 2024 International Conference on Management of Data*, 2024, pp. 120–134.
- [62] Y. Wu, C. Tian, J. Li, H. Sun, K. Tam, L. Li, and C. Xu, “A survey on federated fine-tuning of large language models,” *arXiv preprint arXiv:2503.12016*, 2025.
- [63] T. Che, J. Liu, Y. Zhou, J. Ren, J. Zhou, V. S. Sheng, H. Dai, and D. Dou, “Federated learning of large language models with parameter-efficient prompt tuning and adaptive optimization,” *arXiv preprint arXiv:2310.15080*, 2023.
- [64] J. Zhang, M. Yuan, R. Zhong, P. Luo, H. Zhan, N. Zhang, C. Hu, and X.-Y. Li, “A-vl: Adaptive attention for large vision-language models,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 39, no. 21, 2025, pp. 22 461–22 469.

- [65] Y. Li, H. Jiang, C. Zhang, Q. Wu, X. Luo, S. Ahn, A. H. Abdi, D. Li, J. Gao, Y. Yang *et al.*, “Mminference: Accelerating pre-filling for long-context visual language models via modality-aware permutation sparse attention,” in *Forty-second International Conference on Machine Learning*.
- [66] L. Shen, G. Chen, R. Shao, W. Guan, and L. Nie, “Mome: Mixture of multimodal experts for generalist multimodal large language models,” *Advances in neural information processing systems*, vol. 37, pp. 42 048–42 070, 2024.
- [67] J. Xin, S. Yun, J. Peng, I. Choi, J. L. Ballard, T. Chen, and Q. Long, “I2moe: Interpretable multimodal interaction-aware mixture-of-experts,” *arXiv preprint arXiv:2505.19190*, 2025.
- [68] J. Yu, H. Xiong, L. Zhang, H. Diao, Y. Zhuge, L. Hong, D. Wang, H. Lu, Y. He, and L. Chen, “Llms can evolve continually on modality for backslash x-modal reasoning,” *Advances in Neural Information Processing Systems*, vol. 37, pp. 49 834–49 858, 2024.
- [69] X. Han, H. Nguyen, C. Harris, N. Ho, and S. Saria, “Fusemoe: Mixture-of-experts transformers for fleximodal fusion,” *Advances in Neural Information Processing Systems*, vol. 37, pp. 67 850–67 900, 2024.
- [70] J. Zhang, C. Xiang, H. Huang, H. Xi, J. Zhu, J. Chen *et al.*, “Spargettention: Accurate and training-free sparse attention accelerating any model inference,” in *Forty-second International Conference on Machine Learning*.
- [71] D. Driess, F. Xia, M. S. Sajjadi, C. Lynch, A. Chowdhery, A. Wahid, J. Tompson, Q. Vuong, T. Yu, W. Huang *et al.*, “Palm-e: An embodied multimodal language model,” 2023.
- [72] Y. Shen, K. Song, X. Tan, D. Li, W. Lu, and Y. Zhuang, “Hugginggpt: Solving ai tasks with chatgpt and its friends in hugging face,” *Advances in Neural Information Processing Systems*, vol. 36, pp. 38 154–38 180, 2023.
- [73] Z. Wang, W. Chen, L. Yang, S. Zhou, S. Zhao, H. Zhan, J. Jin, L. Li, Z. Shao, and J. Bu, “Mp-gui: Modality perception with mllms for gui understanding,” in *Proceedings of the Computer Vision and Pattern Recognition Conference*, 2025, pp. 29 711–29 721.
- [74] D. Chen, Y. Huang, S. Wu, J. Tang, L. Chen, Y. Bai, Z. He, C. Wang, H. Zhou, Y. Li *et al.*, “Gui-world: A dataset for gui-oriented multimodal llm-based agents,” *arXiv e-prints*, pp. arXiv-2406, 2024.
- [75] J. S. Park, C. Q. Zou, A. Shaw, B. M. Hill, C. Cai, M. R. Morris, R. Willer, P. Liang, and M. S. Bernstein, “Generative agent simulations of 1,000 people,” *arXiv preprint arXiv:2411.10109*, 2024.
- [76] C. Adornetto, A. Mora, K. Hu, L. I. Garcia, P. Atchade-Adelomou, G. Greco, L. A. A. Pastor, and K. Larson, “Generative agents in agent-based modeling: Overview, validation, and emerging challenges,” *IEEE Transactions on Artificial Intelligence*, 2025.
- [77] B. Ouyang, S. Ye, L. Zeng, T. Qian, J. Li, and X. Chen, “Pluto and charon: A time and memory efficient collaborative edge ai framework for personal llms fine-tuning,” in *Proceedings of the 53rd International Conference on Parallel Processing*, 2024, pp. 762–771.
- [78] X. Zhang, H. Xu, Z. Ba, Z. Wang, Y. Hong, J. Liu, Z. Qin, and K. Ren, “Privacyast: Safeguarding user privacy in tool-using large language model agents,” *IEEE Transactions on Dependable and Secure Computing*, vol. 21, no. 6, pp. 5242–5258, 2024.
- [79] Y. Zhong, S. Liu, J. Chen, J. Hu, Y. Zhu, X. Liu, X. Jin, and H. Zhang, “[DistServe]: Disaggregating prefill and decoding for goodput-optimized large language model serving,” in *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24)*, 2024, pp. 193–210.
- [80] A. Agrawal, A. Agarwal, N. Kedia, J. Mohan, S. Kundu, N. Kwatra, R. Ramjee, and A. Tumanov, “Etalon: holistic performance evaluation framework for llm inference systems,” *arXiv preprint arXiv:2407.07000*, 2024.
- [81] R. Kong, Y. Li, Q. Feng, W. Wang, X. Ye, Y. Ouyang, L. Kong, and Y. Liu, “Swapmoe: Serving off-the-shelf moe-based large language models with tunable memory budget,” *arXiv preprint arXiv:2308.15030*, 2023.
- [82] S. Gao, Y. Chen, and J. Shu, “Fast state restoration in llm serving with hcache,” in *Proceedings of the Twentieth European Conference on Computer Systems*, 2025, pp. 128–143.
- [83] S. Laskaridis, K. Katevas, L. Minto, and H. Haddadi, “Melting point: Mobile evaluation of language transformers,” in *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking*, 2024, pp. 890–907.
- [84] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.
- [85] A. Fang, A. M. Jose, A. Jain, L. Schmidt, A. Toshev, and V. Shankar, “Data filtering networks,” *arXiv preprint arXiv:2309.17425*, 2023.
- [86] A. Chevalier, A. Wettig, A. Ajith, and D. Chen, “Adapting language models to compress contexts,” *arXiv preprint arXiv:2305.14788*, 2023.
- [87] H. Jung and K.-J. Kim, “Discrete prompt compression with reinforcement learning,” *IEEE Access*, 2024.
- [88] H. Wen, Y. Li, G. Liu, S. Zhao, T. Yu, T. J.-J. Li, S. Jiang, Y. Liu, Y. Zhang, and Y. Liu, “Autodroid: Llm-powered task automation in android,” in *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking*, 2024, pp. 543–557.
- [89] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel *et al.*, “Retrieval-augmented generation for knowledge-intensive nlp tasks,” *Advances in neural information processing systems*, vol. 33, pp. 9459–9474, 2020.
- [90] A. Asai, Z. Wu, Y. Wang, A. Sil, and H. Hajishirzi, “Self-rag: Learning to retrieve, generate, and critique through self-reflection,” in *The Twelfth International Conference on Learning Representations*, 2023.
- [91] Z. Jiang, F. F. Xu, L. Gao, Z. Sun, Q. Liu, J. Dwivedi-Yu, Y. Yang, J. Callan, and G. Neubig, “Active retrieval augmented generation,” in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023, pp. 7969–7992.
- [92] W. Shi, S. Min, M. Yasunaga, M. Seo, R. James, M. Lewis, L. Zettlemoyer, and W.-t. Yih, “Replug: Retrieval-augmented black-box language models,” *arXiv preprint arXiv:2301.12652*, 2023.
- [93] N. Shinn, F. Cassano, A. Gopinath, K. Narasimhan, and S. Yao, “Reflexion: Language agents with verbal reinforcement learning,” *Advances in Neural Information Processing Systems*, vol. 36, pp. 8634–8652, 2023.
- [94] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao, “React: Synergizing reasoning and acting in language models,” in *International Conference on Learning Representations (ICLR)*, 2023.
- [95] A. Bulatov, Y. Kuratov, and M. Burtsev, “Recurrent memory transformer,” *Advances in Neural Information Processing Systems*, vol. 35, pp. 11 079–11 091, 2022.
- [96] W. Yu, H. Zhang, X. Pan, K. Ma, H. Wang, and D. Yu, “Chain-of-note: Enhancing robustness in retrieval-augmented language models,” *arXiv preprint arXiv:2311.09210*, 2023.
- [97] L. Gao, X. Ma, J. Lin, and J. Callan, “Precise zero-shot dense retrieval without relevance labels,” in *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2023, pp. 1762–1777.
- [98] Z. Yu, C. Xiong, S. Yu, and Z. Liu, “Augmentation-adapted retriever improves generalization of language models as generic plug-in,” *arXiv preprint arXiv:2305.17331*, 2023.
- [99] A. Zou, Z. Zhang, H. Zhao, and X. Tang, “Generalizable chain-of-thought prompting in mixed-task scenarios with large language models,” *arXiv preprint arXiv:2310.06692*, 2023.
- [100] P. Aggarwal, A. Madaan, Y. Yang *et al.*, “Let’s sample step by step: Adaptive-consistency for efficient reasoning and coding with llms,” *arXiv preprint arXiv:2305.11860*, 2023.
- [101] J. Lanchantin, S. Toshniwal, J. Weston, S. Sukhbaatar *et al.*, “Learning to reason and memorize with self-notes,” *Advances in Neural Information Processing Systems*, vol. 36, pp. 11 891–11 911, 2023.
- [102] L. Gao, A. Madaan, S. Zhou, U. Alon, P. Liu, Y. Yang, J. Callan, and G. Neubig, “Pal: Program-aided language models,” 2023. [Online]. Available: <https://arxiv.org/abs/2211.10435>
- [103] S. Yao, D. Yu, J. Zhao, I. Shafran, T. Griffiths, Y. Cao, and K. Narasimhan, “Tree of thoughts: Deliberate problem solving with large language models,” *Advances in neural information processing systems*, vol. 36, pp. 11 809–11 822, 2023.
- [104] M. Besta, N. Blach, A. Kubicek, R. Gerstenberger, M. Podstawska, L. Gianinazzi, J. Gajda, T. Lehmann, H. Niewiadomski, P. Nyczyk, and T. Hoefler, “Graph of thoughts: Solving elaborate problems with large language models,” *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 16, p. 17682–17690, Mar. 2024. [Online]. Available: <http://dx.doi.org/10.1609/aaai.v38i16.29720>
- [105] A. Madaan, N. Tandon, P. Gupta, S. Hallinan, L. Gao, S. Wiegreffe, U. Alon, N. Dziri, S. Prabhumoye, Y. Yang *et al.*, “Self-refine: Iterative refinement with self-feedback,” *Advances in Neural Information Processing Systems*, vol. 36, pp. 46 534–46 594, 2023.
- [106] D. Guo, D. Yang, H. Zhang, J. Song, R. Zhang, R. Xu, Q. Zhu, S. Ma, P. Wang, X. Bi *et al.*, “Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning,” *arXiv preprint arXiv:2501.12948*, 2025.
- [107] T. D. Le, H. Imai, Y. Negishi, and K. Kawachiya, “Automatic gpu memory management for large neural models in tensorflow,” in *Proceedings of the 2019 ACM SIGPLAN International Symposium on Memory Management*, 2019, pp. 1–13.

- [108] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann *et al.*, “Palm: Scaling language modeling with pathways,” *Journal of Machine Learning Research*, vol. 24, no. 240, pp. 1–113, 2023.
- [109] W. Liu, H. Hu, J. Zhou, Y. Ding, J. Li, J. Zeng, M. He, Q. Chen, B. Jiang, A. Zhou *et al.*, “Mathematical language models: A survey,” *arXiv preprint arXiv:2312.07622*, 2023.
- [110] C. Lou, Z. Jia, Z. Zheng, and K. Tu, “Sparsor is faster and less is more: Efficient sparse attention for long-range transformers,” *arXiv preprint arXiv:2406.16747*, 2024.
- [111] A. Roy, M. Saffar, A. Vaswani, and D. Grangier, “Efficient content-based sparse attention with routing transformers,” *Transactions of the Association for Computational Linguistics*, vol. 9, pp. 53–68, 2021.
- [112] J. Yuan, H. Gao, D. Dai, J. Luo, L. Zhao, Z. Zhang, Z. Xie, Y. Wei, L. Wang, Z. Xiao *et al.*, “Native sparse attention: Hardware-aligned and natively trainable sparse attention,” *arXiv preprint arXiv:2502.11089*, 2025.
- [113] J. Lin, J. Tang, H. Tang, S. Yang, W.-M. Chen, W.-C. Wang, G. Xiao, X. Dang, C. Gan, and S. Han, “Awq: Activation-aware weight quantization for llm compression and acceleration,” 2024. [Online]. Available: <https://arxiv.org/abs/2306.00978>
- [114] J. Wei, S. Cao, T. Cao, L. Ma, L. Wang, Y. Zhang, and M. Yang, “T-mac: Cpu renaissance via table lookup for low-bit llm deployment on edge,” in *Proceedings of the Twentieth European Conference on Computer Systems*, 2025, pp. 278–292.
- [115] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, “Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter,” 2020. [Online]. Available: <https://arxiv.org/abs/1910.01108>
- [116] S. Sun, Y. Cheng, Z. Gan, and J. Liu, “Patient knowledge distillation for bert model compression,” 2019. [Online]. Available: <https://arxiv.org/abs/1908.09355>
- [117] C. Hooper, S. Kim, H. Mohammadzadeh, M. Maheswaran, J. Paik, M. W. Mahoney, K. Keutzer, and A. Gholami, “Squeezed attention: Accelerating long context length llm inference,” *arXiv preprint arXiv:2411.09688*, 2024.
- [118] G. Xiao, J. Tang, J. Zuo, J. Guo, S. Yang, H. Tang, Y. Fu, and S. Han, “Duoattention: Efficient long-context llm inference with retrieval and streaming heads,” *arXiv preprint arXiv:2410.10819*, 2024.
- [119] L. Zhu, X. Wang, Z. Ke, W. Zhang, and R. W. Lau, “Biformer: Vision transformer with bi-level routing attention,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2023, pp. 10 323–10 333.
- [120] G. Ioannides, A. Chadha, and A. Elkins, “Gaussian adaptive attention is all you need: Robust contextual representations across multiple modalities,” *CoRR*, 2024.
- [121] Y. Rao, W. Zhao, B. Liu, J. Lu, J. Zhou, and C.-J. Hsieh, “Dynamicvlt: Efficient vision transformers with dynamic token sparsification,” 2021. [Online]. Available: <https://arxiv.org/abs/2106.02034>
- [122] X. Chen, Y. Cheng, S. Wang, Z. Gan, Z. Wang, and J. Liu, “Earlybert: Efficient bert training via early-bird lottery tickets,” 2021. [Online]. Available: <https://arxiv.org/abs/2101.00063>
- [123] Z. Liu, B. Oguz, C. Zhao, E. Chang, P. Stock, Y. Mehdad, Y. Shi, R. Krishnamoorthi, and V. Chandra, “Llm-qat: Data-free quantization aware training for large language models,” 2023. [Online]. Available: <https://arxiv.org/abs/2305.17888>
- [124] X. Jiao, Y. Yin, L. Shang, X. Jiang, X. Chen, L. Li, F. Wang, and Q. Liu, “Tinybert: Distilling bert for natural language understanding,” 2020. [Online]. Available: <https://arxiv.org/abs/1909.10351>
- [125] Y. Gu, L. Dong, F. Wei, and M. Huang, “Minillm: Knowledge distillation of large language models,” 2024. [Online]. Available: <https://arxiv.org/abs/2306.08543>
- [126] M. Xia, T. Gao, Z. Zeng, and D. Chen, “Sheared llama: Accelerating language model pre-training via structured pruning,” 2024. [Online]. Available: <https://arxiv.org/abs/2310.06694>
- [127] Y. Li, Y. Huang, B. Yang, B. Venkitesh, A. Locatelli, H. Ye, T. Cai, P. Lewis, and D. Chen, “Snapkv: Llm knows what you are looking for before generation,” 2024. [Online]. Available: <https://arxiv.org/abs/2404.14469>
- [128] E. Frantar, S. Ashkboos, T. Hoefer, and D. Alistarh, “Gptq: Accurate post-training quantization for generative pre-trained transformers,” 2023. [Online]. Available: <https://arxiv.org/abs/2210.17323>
- [129] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, “Qlora: Efficient finetuning of quantized llms,” pp. 10 088–10 115, 2023.
- [130] F. Cai, D. Yuan, Z. Yang, and L. Cui, “Edge-llm: A collaborative framework for large language model serving in edge computing,” in *2024 IEEE International Conference on Web Services (ICWS)*. IEEE, 2024, pp. 799–809.
- [131] W. Wang, F. Wei, L. Dong, H. Bao, N. Yang, and M. Zhou, “Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers,” 2020. [Online]. Available: <https://arxiv.org/abs/2002.10957>
- [132] X. Lu, Q. Liu, Y. Xu, A. Zhou, S. Huang, B. Zhang, J. Yan, and H. Li, “Not all experts are equal: Efficient expert pruning and skipping for mixture-of-experts large language models,” *arXiv preprint arXiv:2402.14800*, 2024.
- [133] J. Li, Y. Jiang, Y. Zhu, C. Wang, and H. Xu, “Accelerating distributed {MoE} training and inference with lina,” in *2023 USENIX Annual Technical Conference (USENIX ATC 23)*, 2023, pp. 945–959.
- [134] J. Li, Z. Sun, D. Lin, X. He, Y. Lin, B. Zheng, L. Zeng, R. Zhao, and X. Chen, “Expert-token resonance: Redefining moe routing through affinity-driven active selection,” *arXiv preprint arXiv:2406.00023*, 2024.
- [135] J. Yao, Q. Anthony, A. Shafi, H. Subramoni, and D. K. D. Panda, “Exploiting inter-layer expert affinity for accelerating mixture-of-experts model inference,” in *2024 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE, 2024, pp. 915–925.
- [136] Y. Zhou, T. Lei, H. Liu, N. Du, Y. Huang, V. Zhao, A. M. Dai, Q. V. Le, J. Laudon *et al.*, “Mixture-of-experts with expert choice routing,” *Advances in Neural Information Processing Systems*, vol. 35, pp. 7103–7114, 2022.
- [137] S. Bae, J. Ko, H. Song, and S.-Y. Yun, “Fast and robust early-exiting framework for autoregressive language models with synchronized parallel decoding,” *arXiv preprint arXiv:2310.05424*, 2023.
- [138] T. Schuster, A. Fisch, J. Gupta, M. Dehghani, D. Bahri, V. Tran, Y. Tay, and D. Metzler, “Confident adaptive language modeling,” *Advances in Neural Information Processing Systems*, vol. 35, pp. 17 456–17 472, 2022.
- [139] Y. Chen, X. Pan, Y. Li, B. Ding, and J. Zhou, “Ee-llm: Large-scale training and inference of early-exit large language models with 3d parallelism,” *arXiv preprint arXiv:2312.04916*, 2023.
- [140] L. Del Corro, A. Del Giorno, S. Agarwal, B. Yu, A. Awadallah, and S. Mukherjee, “Skipdecode: Autoregressive skip decoding with batching and caching for efficient llm inference,” *arXiv preprint arXiv:2307.02628*, 2023.
- [141] B.-K. Kim, G. Kim, T.-H. Kim, T. Castells, S. Choi, J. Shin, and H.-K. Song, “Shortened llama: Depth pruning for large language models with comparison of retraining methods,” *arXiv preprint arXiv:2402.02834*, 2024.
- [142] D. Zeng, N. Du, T. Wang, Y. Xu, T. Lei, Z. Chen, and C. Cui, “Learning to skip for language modeling,” *arXiv preprint arXiv:2311.15436*, 2023.
- [143] Y. Jiang, H. Wang, L. Xie, H. Zhao, H. Qian, J. Lui *et al.*, “D-llm: A token adaptive computing resource allocation strategy for large language models,” *Advances in Neural Information Processing Systems*, vol. 37, pp. 1725–1749, 2024.
- [144] R. Hwang, J. Wei, S. Cao, C. Hwang, X. Tang, T. Cao, and M. Yang, “Pre-gated moe: An algorithm-system co-design for fast and scalable mixture-of-expert inference,” in *2024 ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2024, pp. 1018–1031.
- [145] Y. Wei, J. Du, J. Jiang, X. Shi, X. Zhang, D. Huang, N. Xiao, and Y. Lu, “Aptmoe: Affinity-aware pipeline tuning for moe models on bandwidth-constrained gpu nodes,” in *SC24: International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 2024, pp. 1–14.
- [146] S. Fan, X. Jiang, X. Li, X. Meng, P. Han, S. Shang, A. Sun, Y. Wang, and Z. Wang, “Not all layers of llms are necessary during inference,” *arXiv preprint arXiv:2403.02181*, 2024.
- [147] Y. Yang, Z. Cao, and H. Zhao, “Laco: Large language model pruning via layer collapse,” *arXiv preprint arXiv:2402.11187*, 2024.
- [148] G. Liu, C. Li, J. Zhao, C. Zhang, and M. Guo, “Clusterkv: Manipulating llm kv cache in semantic space for recallable compression,” *arXiv preprint arXiv:2412.03213*, 2024.
- [149] C. Xiao, P. Zhang, X. Han, G. Xiao, Y. Lin, Z. Zhang, Z. Liu, and M. Sun, “Inflm: Training-free long-context extrapolation for llms with an efficient context memory,” *arXiv preprint arXiv:2402.04617*, 2024.
- [150] N. Shazeer, “Fast transformer decoding: One write-head is all you need,” *arXiv preprint arXiv:1911.02150*, 2019.
- [151] J. Ainslie, J. Lee-Thorp, M. De Jong, Y. Zemlyanskiy, F. Lebrón, and S. Sanghai, “Gqa: Training generalized multi-query transformer models from multi-head checkpoints,” *arXiv preprint arXiv:2305.13245*, 2023.
- [152] T. Munkhdalai, M. Faruqui, and S. Gopal, “Leave no context behind: Efficient infinite context transformers with infini-attention,” *arXiv preprint arXiv:2404.07143*, vol. 101, 2024.

- [153] X. Zhou, W. Wang, M. Zeng, J. Guo, X. Liu, L. Shen, M. Zhang, and L. Ding, “Dynamickv: Task-aware adaptive kv cache compression for long context llms,” *arXiv preprint arXiv:2412.14838*, 2024.
- [154] Z. Cai, Y. Zhang, B. Gao, Y. Liu, Y. Li, T. Liu, K. Lu, W. Xiong, Y. Dong, J. Hu *et al.*, “Pyramidkv: Dynamic kv cache compression based on pyramidal information funneling,” *arXiv preprint arXiv:2406.02069*, 2024.
- [155] Y. Feng, J. Lv, Y. Cao, X. Xie, and S. K. Zhou, “Ada-kv: Optimizing kv cache eviction by adaptive budget allocation for efficient llm inference,” *arXiv preprint arXiv:2407.11550*, 2024.
- [156] Y. Fu, Z. Cai, A. Asi, W. Xiong, Y. Dong, and W. Xiao, “Not all heads matter: A head-level kv cache compression method with integrated retrieval and reasoning,” *arXiv preprint arXiv:2410.19258*, 2024.
- [157] Z. Liu, J. Yuan, H. Jin, S. Zhong, Z. Xu, V. Braverman, B. Chen, and X. Hu, “Kivi: A tuning-free asymmetric 2bit quantization for kv cache,” *arXiv preprint arXiv:2402.02750*, 2024.
- [158] D. Liu, M. Chen, B. Lu, H. Jiang, Z. Han, Q. Zhang, Q. Chen, C. Zhang, B. Ding, K. Zhang *et al.*, “Retrievalattention: Accelerating long-context llm inference via vector retrieval,” *arXiv preprint arXiv:2409.10516*, 2024.
- [159] Z. Wan, H. Shen, X. Wang, C. Liu, Z. Mai, and M. Zhang, “Meda: Dynamic kv cache allocation for efficient multimodal long-context inference,” *arXiv preprint arXiv:2502.17599*, 2025.
- [160] H. Huang, S. Zhong, Z. Zhang, S. Li, D. Niu, H. Zheng, R. Wang, and M. Li, “Hd-moe: Hybrid and dynamic parallelism for mixture-of-expert llms with 3d near-memory processing,” *arXiv preprint arXiv:2509.09420*, 2025.
- [161] J. Yan, J. Liu, H. Xu, and L. Huang, “Accelerating mixture-of-expert inference with adaptive expert split mechanism,” *arXiv preprint arXiv:2509.08342*, 2025.
- [162] Z. Jia, O. Padon, J. Thomas, T. Warszawski, M. Zaharia, and A. Aiken, “Taso: optimizing deep learning computation with automatic generation of graph substitutions,” in *Proceedings of the 27th ACM Symposium on Operating Systems Principles*, 2019, pp. 47–62.
- [163] W. Kwon, Z. Li, S. Zhuang, Y. Sheng, L. Zheng, C. H. Yu, J. Gonzalez, H. Zhang, and I. Stoica, “Efficient memory management for large language model serving with pagedattention,” in *Proceedings of the 29th symposium on operating systems principles*, 2023, pp. 611–626.
- [164] Q. Su, W. Zhao, X. Li, M. Andoorveedu, C. Jiang, Z. Zhu, K. Song, C. Giannoula, and G. Pekhimenko, “Seesaw: High-throughput llm inference via model re-sharding,” *arXiv preprint arXiv:2503.06433*, 2025.
- [165] G.-I. Yu, J. S. Jeong, G.-W. Kim, S. Kim, and B.-G. Chun, “Orca: A distributed serving system for {Transformer-Based} generative models,” in *16th USENIX Symposium on Operating Systems Design and Implementation (OSDI 22)*, 2022, pp. 521–538.
- [166] R. Y. Aminabadi, S. Rajbhandari, A. A. Awan, C. Li, D. Li, E. Zheng, O. Ruwase, S. Smith, M. Zhang, J. Rasley *et al.*, “Deepspeed-inference: enabling efficient inference of transformer models at unprecedented scale,” in *SC22: International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 2022, pp. 1–15.
- [167] A. Sun, W. Zhao, X. Han, C. Yang, Z. Liu, C. Shi, and M. Sun, “Burstattention: An efficient distributed attention framework for extremely long sequences,” *arXiv preprint arXiv:2403.09347*, 2024.
- [168] B. Butler, S. Yu, A. Mazaheri, and A. Jannesari, “Pipeinfer: Accelerating llm inference using asynchronous pipelined speculation,” in *SC24: International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 2024, pp. 1–19.
- [169] X. Miao, G. Oliaro, Z. Zhang, X. Cheng, Z. Wang, R. Y. Y. Wong, Z. Chen, D. Arfeen, R. Abhyankar, and Z. Jia, “Specinfer: Accelerating generative llm serving with speculative inference and token tree verification,” *arXiv preprint arXiv:2305.09781*, vol. 1, no. 2, p. 4, 2023.
- [170] H. Du, S. Wu, A. Kharlamova, N. Guan, and C. J. Xue, “Flexinfer: Breaking memory constraint via flexible and efficient offloading for on-device llm inference,” in *Proceedings of the 5th Workshop on Machine Learning and Systems*, 2025, pp. 56–65.
- [171] S. Cao, S. Liu, T. Griggs, P. Schafhalter, X. Liu, Y. Sheng, J. E. Gonzalez, M. Zaharia, and I. Stoica, “Moe-lightning: High-throughput moe inference on memory-constrained gpus,” in *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 1*, 2025, pp. 715–730.
- [172] Y. Song, Z. Mi, H. Xie, and H. Chen, “Powerinfer: Fast large language model serving with a consumer-grade gpu,” in *Proceedings of the ACM SIGOPS 30th Symposium on Operating Systems Principles*, 2024, pp. 590–606.
- [173] X. Zhao, B. Jia, H. Zhou, Z. Liu, S. Cheng, and Y. You, “Hetegen: Heterogeneous parallel inference for large language models on resource-constrained devices,” *arXiv preprint arXiv:2403.01164*, 2024.
- [174] J. Shah, G. Bikshandi, Y. Zhang, V. Thakkar, P. Ramani, and T. Dao, “Flashattention-3: Fast and accurate attention with asynchrony and low-precision,” *Advances in Neural Information Processing Systems*, vol. 37, pp. 68 658–68 685, 2024.
- [175] Y. Li, S. Jiang, B. Hu, L. Wang, W. Zhong, W. Luo, L. Ma, and M. Zhang, “Uni-moe: Scaling unified multimodal llms with mixture of experts,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2025.
- [176] P. Patel, E. Choukse, C. Zhang, A. Shah, Í. Goiri, S. Maleki, and R. Bianchini, “Splitwise: Efficient generative llm inference using phase splitting,” in *2024 ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2024, pp. 118–132.
- [177] Y. Sheng, L. Zheng, B. Yuan, Z. Li, M. Ryabinin, B. Chen, P. Liang, C. Ré, I. Stoica, and C. Zhang, “Flexgen: High-throughput generative inference of large language models with a single gpu,” in *International Conference on Machine Learning*. PMLR, 2023, pp. 31 094–31 116.
- [178] G. Neubig, C. Dyer, Y. Goldberg, A. Matthews, W. Ammar, A. Anastasopoulos, M. Ballesteros, D. Chiang, D. Clothiaux, T. Cohn *et al.*, “Dynet: The dynamic neural network toolkit,” *arXiv preprint arXiv:1701.03980*, 2017.
- [179] M. Looks, M. Herreshoff, D. Hutchins, and P. Norvig, “Deep learning with dynamic computation graphs,” 2017. [Online]. Available: <https://arxiv.org/abs/1702.02181>
- [180] J. Raiman, “Dali: Lazy compilation of dynamic computation graphs,” in *Workshop on Systems for Machine Learning and Open Source Software at NeurIPS*, 2018.
- [181] S. Rajbhandari, C. Li, Z. Yao, M. Zhang, R. Y. Aminabadi, A. A. Awan, J. Rasley, and Y. He, “Deepspeed-moe: Advancing mixture-of-experts inference and training to power next-generation ai scale,” in *International conference on machine learning*. PMLR, 2022, pp. 18 332–18 346.
- [182] S. Mittal and J. S. Vetter, “A survey of cpu-gpu heterogeneous computing techniques,” *ACM Comput. Surv.*, vol. 47, no. 4, Jul. 2015. [Online]. Available: <https://doi.org/10.1145/2788396>
- [183] M. Rhu, N. Gimelshein, J. Clemons, A. Zulfiqar, and S. W. Keckler, “vdnn: Virtualized deep neural networks for scalable, memory-efficient neural network design,” in *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 2016, pp. 1–13.
- [184] T. Hoefer, T. Bonato, D. De Sensi, S. Di Girolamo, S. Li, M. Hedges, D. Goel, M. Castro, and S. Scott, “Hammingmesh: A network topology for large-scale deep learning,” *Commun. ACM*, vol. 67, no. 12, p. 97–105, Nov. 2024. [Online]. Available: <https://doi.org/10.1145/3623490>
- [185] M. Davies, N. Crago, K. Sankaralingam, and C. Kozyrakis, “Efficient llm inference: Bandwidth, compute, synchronization, and capacity are all you need,” 2025. [Online]. Available: <https://arxiv.org/abs/2507.14397>
- [186] S. Yun, S. Park, H. Nam, Y. Lee, G. Lee, K. Kyung, S. Kim, N. S. Kim, J. Kim, H. Kim, J. Cho, S. Baek, and J. H. Ahn, “The new llm bottleneck: A systems perspective on latent attention and mixture-of-experts,” 2025. [Online]. Available: <https://arxiv.org/abs/2507.15465>
- [187] W. Lee, J. Lee, J. Seo, and J. Sim, “[InfiniGen]: Efficient generative inference of large language models with dynamic {KV} cache management,” in *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24)*, 2024, pp. 155–172.
- [188] C. Jiang, L. Gao, H. E. Zarch, and M. Annavaram, “Kvpr: Efficient llm inference with i/o-aware kv cache partial recomputation,” *arXiv preprint arXiv:2411.17089*, 2024.
- [189] Y. Xu, Z. Mao, X. Mo, S. Liu, and I. Stoica, “Pie: Pooling cpu memory for llm inference,” *arXiv preprint arXiv:2411.09317*, 2024.
- [190] J. Zhao, B. Wan, C. Wu, Y. Peng, and H. Lin, “Poster: Llm-pq: Serving llm on heterogeneous clusters with phase-aware partition and adaptive quantization,” in *Proceedings of the 29th ACM SIGPLAN Annual Symposium on Principles and Practice of Parallel Programming*, 2024, pp. 460–462.
- [191] J. Stojkovic, C. Zhang, Í. Goiri, J. Torrellas, and E. Choukse, “Dynamollm: Designing llm inference clusters for performance and energy efficiency,” in *2025 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 2025, pp. 1348–1362.
- [192] H. M. Chen, W. Luk, K. F. C. Yiu, R. Li, K. Mishchenko, S. I. Venieris, and H. Fan, “Hardware-aware parallel prompt decoding for memory-efficient acceleration of llm inference,” *arXiv preprint arXiv:2405.18628*, 2024.

- [193] X. Jiang, Y. Zhou, S. Cao, I. Stoica, and M. Yu, “Neo: Saving gpu memory crisis with cpu offloading for online llm inference,” *arXiv preprint arXiv:2411.01142*, 2024.
- [194] C. Luo, Z. Cai, H. Sun, J. Xiao, B. Yuan, W. Xiao, J. Hu, J. Zhao, B. Chen, and A. Anandkumar, “Headinfer: Memory-efficient llm inference by head-wise offloading,” *arXiv preprint arXiv:2502.12574*, 2025.
- [195] H. Jang, S. Noh, C. Shin, J. Jung, J. Song, and J. Lee, “Inf² 2: High-throughput generative inference of large language models using near-storage processing,” *arXiv preprint arXiv:2502.09921*, 2025.
- [196] Y. Wen, N. Jain, J. Kirchenbauer, M. Goldblum, J. Geiping, and T. Goldstein, “Hard prompts made easy: Gradient-based discrete optimization for prompt tuning and discovery,” *Advances in Neural Information Processing Systems*, vol. 36, pp. 51 008–51 025, 2023.
- [197] M. Deng, J. Wang, C.-P. Hsieh, Y. Wang, H. Guo, T. Shu, M. Song, E. P. Xing, and Z. Hu, “Rlprompt: Optimizing discrete text prompts with reinforcement learning,” *arXiv preprint arXiv:2205.12548*, 2022.
- [198] S. Bai, J. Zhang, S. Guo, S. Li, J. Guo, J. Hou, T. Han, and X. Lu, “Diprompt: Disentangled prompt tuning for multiple latent domain generalization in federated learning,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 27 284–27 293.
- [199] J. Ye, Z. Wu, J. Feng, T. Yu, and L. Kong, “Compositional exemplars for in-context learning,” in *International Conference on Machine Learning*. PMLR, 2023, pp. 39 818–39 833.
- [200] B. Wang, G. Li, and Y. Li, “Enabling conversational interaction with mobile ui using large language models,” in *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, 2023, pp. 1–17.
- [201] J. Yao, Y. Liu, Z. Dong, M. Guo, H. Hu, K. Keutzer, L. Du, D. Zhou, and S. Zhang, “Promptcot: Align prompt distribution via adapted chain-of-thought,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 7027–7037.
- [202] X. Yuan, C. Shen, S. Yan, X. Zhang, L. Xie, W. Wang, R. Guan, Y. Wang, and J. Ye, “Instance-adaptive zero-shot chain-of-thought prompting,” *Advances in Neural Information Processing Systems*, vol. 37, pp. 125 469–125 486, 2024.
- [203] M. Zawalski, W. Chen, K. Pertsch, O. Mees, C. Finn, and S. Levine, “Robotic control via embodied chain-of-thought reasoning,” *arXiv preprint arXiv:2407.08693*, 2024.
- [204] H. Chae, Y. Song, K. T.-i. Ong, T. Kwon, M. Kim, Y. Yu, D. Lee, D. Kang, and J. Yeo, “Dialogue chain-of-thought distillation for commonsense-aware conversational agents,” *arXiv preprint arXiv:2310.09343*, 2023.
- [205] S. He, L. Ding, D. Dong, M. Zhang, and D. Tao, “Sparseadapter: An easy approach for improving the parameter-efficiency of adapters,” *arXiv preprint arXiv:2210.04284*, 2022.
- [206] R. Karimi Mahabadi, J. Henderson, and S. Ruder, “Compacter: Efficient low-rank hypercomplex adapter layers,” *Advances in Neural Information Processing Systems*, vol. 34, pp. 1022–1035, 2021.
- [207] Y. Zhuang, Z. Zheng, F. Wu, and G. Chen, “Litemoe: Customizing on-device llm serving via proxy submodel tuning,” in *Proceedings of the 22nd ACM Conference on Embedded Networked Sensor Systems*, 2024, pp. 521–534.
- [208] N. Velingker, J. Liu, A. Sethi, W. Dodds, Z. Xu, S. Dutta, M. Naik, and E. Wong, “Clam: Unifying finetuning, quantization, and pruning by chaining llm adapter modules,” in *Workshop on Efficient Systems for Foundation Models II@ ICML2024*.
- [209] Y. Yang, K. Zhen, E. Banijamal, A. Mouchtaris, and Z. Zhang, “Adazeta: Adaptive zeroth-order tensor-train adaption for memory-efficient large language models fine-tuning,” *arXiv preprint arXiv:2406.18060*, 2024.
- [210] X. L. Li and P. Liang, “Prefix-tuning: Optimizing continuous prompts for generation,” *arXiv preprint arXiv:2101.00190*, 2021.
- [211] X. Liu, K. Ji, Y. Fu, W. L. Tam, Z. Du, Z. Yang, and J. Tang, “P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks,” *arXiv preprint arXiv:2110.07602*, 2021.
- [212] R. Qin, P. Ren, Z. Yan, L. Liu, D. Liu, A. Nassereldine, J. Xiong, K. Ni, S. Hu, and Y. Shi, “Nvcim-pt: An nvcim-assisted prompt tuning framework for edge llms,” *arXiv preprint arXiv:2411.08244*, 2024.
- [213] H. Liu, D. Tam, M. Muqeeth, J. Mohta, T. Huang, M. Bansal, and C. A. Raffel, “Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning,” *Advances in Neural Information Processing Systems*, vol. 35, pp. 1950–1965, 2022.
- [214] X. Lu, F. Brahman, P. West, J. Jang, K. Chandu, A. Ravichander, L. Qin, P. Ammanabrolu, L. Jiang, S. Ramnath *et al.*, “Inference-time policy adapters (ipa): Tailoring extreme-scale lms without fine-tuning,” *arXiv preprint arXiv:2305.15065*, 2023.
- [215] N. Lawton, A. Kumar, G. Thattai, A. Galstyan, and G. V. Steeg, “Neural architecture search for parameter-efficient fine-tuning of large pre-trained language models,” *arXiv preprint arXiv:2305.16597*, 2023.
- [216] R. Xu, F. Luo, Z. Zhang, C. Tan, B. Chang, S. Huang, and F. Huang, “Raise a child in large language model: Towards effective and generalizable fine-tuning,” *arXiv preprint arXiv:2109.05687*, 2021.
- [217] A. Ansell, E. M. Ponti, A. Korhonen, and I. Vulić, “Composable sparse fine-tuning for cross-lingual transfer,” *arXiv preprint arXiv:2110.07560*, 2021.
- [218] B. Liao, Y. Meng, and C. Monz, “Parameter-efficient fine-tuning without introducing new latency,” *arXiv preprint arXiv:2305.16742*, 2023.
- [219] D. Vucetic, M. Tayaranian, M. Ziaeefard, J. J. Clark, B. H. Meyer, and W. J. Gross, “Efficient fine-tuning of bert models on the edge,” in *2022 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2022, pp. 1838–1842.
- [220] M. Kowsher, T. Esmaeilbeig, C.-N. Yu, M. Soltanalian, and N. Yousefi, “Rocoft: Efficient finetuning of large language models with row-column updates,” *arXiv preprint arXiv:2410.10075*, 2024.
- [221] M. Valipour, M. Rezagholizadeh, I. Kobyzhev, and A. Ghodsi, “Dylora: Parameter efficient tuning of pre-trained models using dynamic search-free low-rank adaptation,” *arXiv preprint arXiv:2210.07558*, 2022.
- [222] Q. Zhang, M. Chen, A. Bukharin, N. Karampatziakis, P. He, Y. Cheng, W. Chen, and T. Zhao, “Adalora: Adaptive budget allocation for parameter-efficient fine-tuning,” *arXiv preprint arXiv:2303.10512*, 2023.
- [223] L. Gao, A. Ziashahabi, Y. Niu, S. Avestimehr, and M. Annavarapu, “Enabling resource-efficient on-device fine-tuning of llms using only inference engines,” *arXiv preprint arXiv:2409.15520*, 2024.
- [224] W. Zhao, Y. Huang, X. Han, Z. Liu, Z. Zhang, K. Li, C. Chen, T. Yang, and M. Sun, “Ca-lora: Adapting existing lora for compressed llms to enable efficient multi-tasking on personal devices,” *arXiv preprint arXiv:2307.07705*, 2023.
- [225] J. Liu, G. Xiao, K. Li, J. D. Lee, S. Han, T. Dao, and T. Cai, “Bitdelta: Your fine-tune may only be worth one bit,” *Advances in Neural Information Processing Systems*, vol. 37, pp. 13 579–13 600, 2024.
- [226] Z. Gao, Q. Wang, A. Chen, Z. Liu, B. Wu, L. Chen, and J. Li, “Parameter-efficient fine-tuning with discrete fourier transform,” pp. 14 884–14 901, 2024.
- [227] F. Meng, Z. Wang, and M. Zhang, “Pissa: Principal singular values and singular vectors adaptation of large language models,” *Advances in Neural Information Processing Systems*, vol. 37, pp. 121 038–121 072, 2024.
- [228] J. Chen, A. Zhang, X. Shi, M. Li, A. Smola, and D. Yang, “Parameter-efficient fine-tuning design spaces,” *arXiv preprint arXiv:2301.01821*, 2023.
- [229] H. Zhou, X. Wan, I. Vulić, and A. Korhonen, “Autopeft: Automatic configuration search for parameter-efficient fine-tuning,” *Transactions of the Association for Computational Linguistics*, vol. 12, pp. 525–542, 2024.
- [230] B. Zi, X. Qi, L. Wang, J. Wang, K.-F. Wong, and L. Zhang, “Delta-lora: Fine-tuning high-rank parameters with the delta of low-rank matrices,” *arXiv preprint arXiv:2309.02411*, 2023.
- [231] G. Oliaro, X. Miao, X. Cheng, V. Kada, R. Gao, Y. Huang, R. Delacourt, A. Yang, Y. Wang, M. Wu *et al.*, “Flexllm: A system for co-serving large language model inference and parameter-efficient finetuning,” *arXiv preprint arXiv:2402.18789*, 2024.
- [232] M. Zhang, H. Chen, C. Shen, Z. Yang, L. Ou, X. Yu, and B. Zhuang, “Loraprune: Structured pruning meets low-rank parameter-efficient fine-tuning,” *arXiv preprint arXiv:2305.18403*, 2023.
- [233] B. Zhao, H. Hajishirzi, and Q. Cao, “Apt: Adaptive pruning and tuning pretrained language models for efficient training and inference,” *arXiv preprint arXiv:2401.12200*, 2024.
- [234] Y. J. Cho, L. Liu, Z. Xu, A. Fahrezi, and G. Joshi, “Heterogeneous lora for federated fine-tuning of on-device foundation models,” *arXiv preprint arXiv:2401.06432*, 2024.
- [235] Y. Xu, L. Xie, X. Gu, X. Chen, H. Chang, H. Zhang, Z. Chen, X. Zhang, and Q. Tian, “Qa-lora: Quantization-aware low-rank adaptation of large language models,” *arXiv preprint arXiv:2309.14717*, 2023.
- [236] J. Yu, K. Prabhu, Y. Urman, R. M. Radway, E. Han, and P. Raina, “8-bit transformer inference and fine-tuning for edge accelerators,” in *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*, 2024, pp. 5–21.
- [237] C. Lee, J.-g. Jin, Y. Cho, and E. Park, “Qeft: Quantization for efficient fine-tuning of llms,” *arXiv preprint arXiv:2410.08661*, 2024.

- [238] J. Phang, Y. Mao, P. He, and W. Chen, "Hypertuning: Toward adapting large language models without back-propagation," in *International Conference on Machine Learning*. PMLR, 2023, pp. 27 854–27 875.
- [239] Y.-L. Sung, J. Cho, and M. Bansal, "Lst: Ladder side-tuning for parameter and memory efficient transfer learning," *Advances in Neural Information Processing Systems*, vol. 35, pp. 12 991–13 005, 2022.
- [240] S. Malladi, T. Gao, E. Nichani, A. Damian, J. D. Lee, D. Chen, and S. Arora, "Fine-tuning language models with just forward passes," *Advances in Neural Information Processing Systems*, vol. 36, pp. 53 038–53 075, 2023.
- [241] J. Zhao, Z. Zhang, B. Chen, Z. Wang, A. Anandkumar, and Y. Tian, "Galore: Memory-efficient llm training by gradient low-rank projection," *arXiv preprint arXiv:2403.03507*, 2024.
- [242] H. Jiang, Q. Wu, C.-Y. Lin, Y. Yang, and L. Qiu, "Llmlingua: Compressing prompts for accelerated inference of large language models," *arXiv preprint arXiv:2310.05736*, 2023.
- [243] Y. Zhang, S. Feng, and C. Tan, "Active example selection for in-context learning," *arXiv preprint arXiv:2211.04486*, 2022.
- [244] Y. Shao, L. Li, J. Dai, and X. Qiu, "Character-llm: A trainable agent for role-playing," *arXiv preprint arXiv:2310.10158*, 2023.
- [245] A. Fourney, G. Bansal, H. Mozannar, C. Tan, E. Salinas, F. Niedtner, G. Proebsting, G. Bassman, J. Gerrits, J. Alber *et al.*, "Magenticon: A generalist multi-agent system for solving complex tasks," *arXiv preprint arXiv:2411.04468*, 2024.
- [246] L. Li, Z. Chen, G. Chen, Y. Zhang, Y. Su, E. Xing, and K. Zhang, "Confidence matters: Revisiting intrinsic self-correction capabilities of large language models," *arXiv preprint arXiv:2402.12563*, 2024.
- [247] Z. Gou, Z. Shao, Y. Gong, Y. Shen, Y. Yang, N. Duan, and W. Chen, "Critic: Large language models can self-correct with tool-interactive critiquing," *arXiv preprint arXiv:2305.11738*, 2023.
- [248] R. Qin, J. Xia, Z. Jia, M. Jiang, A. Abbasi, P. Zhou, J. Hu, and Y. Shi, "Enabling on-device large language model personalization with self-supervised data selection and synthesis," in *Proceedings of the DAC*, 2024, pp. 1–6.
- [249] J. S. Smith, L. Volkov, S. Halbe, V. Gutta, R. Feris, Z. Kira, and L. Karlinsky, "Adaptive memory replay for continual learning," in *Proceedings of the CVPR*, 2024, pp. 3605–3615.
- [250] A. Modarressi, A. Imani, M. Fayyaz, and H. Schütze, "Ret-llm: Towards a general read-write memory for large language models," *arXiv preprint arXiv:2305.14322*, 2023.
- [251] C. Hu, J. Fu, C. Du, S. Luo, J. Zhao, and H. Zhao, "Chatdb: Augmenting llms with databases as their symbolic memory," *arXiv preprint arXiv:2306.03901*, 2023.
- [252] J. Chen, R. Zhang, J. Guo, Y. Liu, Y. Fan, and X. Cheng, "Corpusbrain: Pre-train a generative retrieval model for knowledge-intensive language tasks," in *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, 2022, pp. 191–200.
- [253] J. Lu, S. An, M. Lin, G. Pergola, Y. He, D. Yin, X. Sun, and Y. Wu, "Memochat: Tuning llms to use memos for consistent long-range open-domain conversation," *arXiv preprint arXiv:2308.08239*, 2023.
- [254] Z. Jiang, L. Zhong, M. Sun, J. Xu, R. Sun, H. Cai, S. Luo, and Z. Zhang, "Efficient knowledge infusion via kg-llm alignment," *arXiv preprint arXiv:2406.03746*, 2024.
- [255] Y. Wen, Z. Wang, and J. Sun, "Mindmap: Knowledge graph prompting sparks graph of thoughts in large language models," *arXiv preprint arXiv:2308.09729*, 2023.
- [256] D. Sanmartin, "Kg-rag: Bridging the gap between knowledge and creativity," *arXiv preprint arXiv:2405.12035*, 2024.
- [257] V. Mavroudis, "Langchain," 2024.
- [258] B. Zirnstein, "Extended context for instructgpt with llamaindex," 2023.
- [259] G. Cui, L. Yuan, N. Ding, G. Yao, B. He, W. Zhu, Y. Ni, G. Xie, R. Xie, Y. Lin *et al.*, "Ultrafeedback: Boosting language models with scaled ai feedback," *arXiv preprint arXiv:2310.01377*, 2023.
- [260] G. Dai, S. Jiang, T. Cao, Y. Li, Y. Yang, R. Tan, M. Li, and L. Qiu, "Advancing mobile gui agents: A verifier-driven approach to practical deployment," *arXiv preprint arXiv:2503.15937*, 2025.
- [261] Y.-F. Zhang, X. Lu, X. Hu, C. Fu, B. Wen, T. Zhang, C. Liu, K. Jiang, K. Chen, K. Tang *et al.*, "R1-reward: Training multimodal reward model through stable reinforcement learning," *arXiv preprint arXiv:2505.02835*, 2025.
- [262] R. Rafailov, A. Sharma, E. Mitchell, C. D. Manning, S. Ermon, and C. Finn, "Direct preference optimization: Your language model is secretly a reward model," *Advances in Neural Information Processing Systems*, vol. 36, pp. 53 728–53 741, 2023.
- [263] M. Wulfmeier, M. Bloesch, N. Vieillard, A. Ahuja, J. Bornschein, S. Huang, A. Sokolov, M. Barnes, G. Desjardins, A. Bewley *et al.*, "Imitating language via scalable inverse reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 37, pp. 90 714–90 735, 2024.
- [264] Y. Duan, Q. Zhang, and R. Xu, "Prompting multi-modal tokens to enhance end-to-end autonomous driving imitation learning with llms," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 6798–6805.
- [265] D. Han, T. McInroe, A. Jolley, S. V. Albrecht, P. Bell, and A. Storkey, "Llm-personalize: Aligning llm planners with human preferences via reinforced self-training for housekeeping robots," *arXiv preprint arXiv:2404.14285*, 2024.
- [266] R. Schumann, W. Zhu, W. Feng, T.-J. Fu, S. Riezler, and W. Y. Wang, "Velma: Verbalization embodiment of llm agents for vision and language navigation in street view," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 17, 2024, pp. 18 924–18 933.
- [267] L. Sun, Z. Tao, Y. Li, and H. Arakawa, "Oda: Observation-driven agent for integrating llms and knowledge graphs," *arXiv preprint arXiv:2404.07677*, 2024.
- [268] H. Ma, T. Hu, Z. Pu, L. Boyin, X. Ai, Y. Liang, and M. Chen, "Coevolving with the other you: Fine-tuning llm with sequential cooperative multi-agent reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 37, pp. 15 497–15 525, 2024.
- [269] D. Zhang, L. Chen, S. Zhang, H. Xu, Z. Zhao, and K. Yu, "Large language models are semi-parametric reinforcement learning agents," *Advances in Neural Information Processing Systems*, vol. 36, pp. 78 227–78 239, 2023.
- [270] P. Putta, E. Mills, N. Garg, S. Motwani, C. Finn, D. Garg, and R. Rafailov, "Agent q: Advanced reasoning and learning for autonomous ai agents," *arXiv preprint arXiv:2408.07199*, 2024.
- [271] Y. Yu, Z. Yao, H. Li, Z. Deng, Y. Jiang, Y. Cao, Z. Chen, J. Suchow, Z. Cui, R. Liu *et al.*, "Fincon: A synthesized llm multi-agent system with conceptual verbal reinforcement for enhanced financial decision making," *Advances in Neural Information Processing Systems*, vol. 37, pp. 137 010–137 045, 2024.
- [272] T. Carta, C. Romac, T. Wolf, S. Lamprier, O. Sigaud, and P.-Y. Oudeyer, "Grounding large language models in interactive environments with online reinforcement learning," in *International Conference on Machine Learning*. PMLR, 2023, pp. 3676–3713.
- [273] S. Rajbhandari, O. Ruwase, J. Rasley, S. Smith, and Y. He, "Zero-infinity: Breaking the gpu memory wall for extreme scale deep learning," in *Proceedings of the international conference for high performance computing, networking, storage and analysis*, 2021, pp. 1–14.
- [274] J. Zhang, S. Ma, P. Liu, and J. Yuan, "Coop: Memory is not a commodity," *Advances in Neural Information Processing Systems*, vol. 36, pp. 49 870–49 882, 2023.
- [275] S. Chen, Z. Wang, Z. Guan, Y. Liu, and P. B. Gibbons, "Practical offloading for fine-tuning llm on commodity gpu via learned sparse projectors," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 39, no. 22, 2025, pp. 23 614–23 622.
- [276] Y. Kim, H. Lim, and D. Han, "Scaling beyond the gpu memory limit for large mixture-of-experts model training," in *Forty-first International Conference on Machine Learning*, 2024.
- [277] H. Jang, J. Song, J. Jung, J. Park, Y. Kim, and J. Lee, "Smart-infinity: Fast large language model training using near-storage processing on a real system," in *2024 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. IEEE, 2024, pp. 345–360.
- [278] Y. Hu, J. Zhang, X. Chen, Z. Zhao, C. Li, and H. Chen, "Lors: Efficient low-rank adaptation for sparse large language model," *arXiv preprint arXiv:2501.08582*, 2025.
- [279] T. Wang, X. Chen, K. Li, T. Cao, J. Ren, and Y. Zhang, "Lemo: Enabling less token involvement for more context fine-tuning," *arXiv preprint arXiv:2501.09767*, 2025.
- [280] L. Zhu, L. Hu, J. Lin, W.-M. Chen, W.-C. Wang, C. Gan, and S. Han, "Pockengine: Sparse and efficient fine-tuning in a pocket," in *Proceedings of the 56th Annual IEEE/ACM International Symposium on Microarchitecture*, 2023, pp. 1381–1394.
- [281] J. Fang, Z. Zhu, S. Li, H. Su, Y. Yu, J. Zhou, and Y. You, "Parallel training of pre-trained models via chunk-based dynamic memory management," *IEEE Transactions on Parallel and Distributed Systems*, vol. 34, no. 1, pp. 304–315, 2023.
- [282] D. Cai, Y. Wu, S. Wang, F. X. Lin, and M. Xu, "Efficient federated learning for modern nlp," in *Proceedings of the 29th Annual International Conference on Mobile Computing and Networking*, 2023, pp. 1–16.

- [283] W. Kuang, B. Qian, Z. Li, D. Chen, D. Gao, X. Pan, Y. Xie, Y. Li, B. Ding, and J. Zhou, “Federatedscope-llm: A comprehensive package for fine-tuning large language models in federated learning,” in *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2024, pp. 5260–5271.
- [284] J. Sun, Z. Xu, H. Yin, D. Yang, D. Xu, Y. Chen, and H. R. Roth, “Fedbpt: Efficient federated black-box prompt tuning for large language models,” *arXiv preprint arXiv:2310.01467*, 2023.
- [285] Z. Fang, Z. Lin, Z. Chen, X. Chen, Y. Gao, and Y. Fang, “Automated federated pipeline for parameter-efficient fine-tuning of large language models,” *arXiv preprint arXiv:2404.06448*, 2024.
- [286] G. Kim, J. Yoo, and S. Kang, “Efficient federated learning with pre-trained large language model using several adapter mechanisms,” *Mathematics*, vol. 11, no. 21, p. 4479, 2023.
- [287] S. Ghiasvand, M. Alizadeh, and R. Pedarsani, “Decentralized low-rank fine-tuning of large language models,” *arXiv preprint arXiv:2501.15361*, 2025.
- [288] S. Wang, J. Liu, H. Xu, J. Yan, and X. Gao, “Efficient federated fine-tuning of large language models with layer dropout,” *arXiv preprint arXiv:2503.10217*, 2025.
- [289] Y. Zhao, A. Gu, R. Varma, L. Luo, C.-C. Huang, M. Xu, L. Wright, H. Shojanazeri, M. Ott, S. Shleifer *et al.*, “Pytorch fsdp: experiences on scaling fully sharded data parallel,” *arXiv preprint arXiv:2304.11277*, 2023.
- [290] A. Borzunov, D. Baranchuk, T. Dettmers, M. Ryabinin, Y. Belkada, A. Chumachenko, P. Samygin, and C. Raffel, “Petals: Collaborative inference and fine-tuning of large models,” *arXiv preprint arXiv:2209.01188*, 2022.
- [291] Z. Lin, Y. Zhang, Z. Chen, Z. Fang, X. Chen, P. Vepakomma, W. Ni, J. Luo, and Y. Gao, “Hsplitllora: A heterogeneous split parameter-efficient fine-tuning framework for large language models,” *arXiv preprint arXiv:2505.02795*, 2025.
- [292] S. Shi, X. Pan, X. Chu, and B. Li, “Pipemoe: Accelerating mixture-of-experts through adaptive pipelining,” in *IEEE INFOCOM 2023-IEEE Conference on Computer Communications*. IEEE, 2023, pp. 1–10.
- [293] Z. Li, S. Wu, L. Li, and S. Zhang, “Energy-efficient split learning for fine-tuning large language models in edge networks,” *IEEE Networking Letters*, 2025.
- [294] S. Zhang, G. Cheng, Z. Li, and W. Wu, “Splitllm: Hierarchical split learning for large language model over wireless network,” *arXiv preprint arXiv:2501.13318*, 2025.
- [295] Z. Xu, K. D. Nguyen, P. Mukherjee, S. Bagchi, S. Chatterji, Y. Liang, and Y. Li, “Learning to inference adaptively for multimodal large language models,” *arXiv preprint arXiv:2503.10905*, 2025.
- [296] B. Mustafa, C. Riquelme, J. Puigcerver, R. Jenatton, and N. Houlsby, “Multimodal contrastive learning with limoe: the language-image mixture of experts,” *Advances in Neural Information Processing Systems*, vol. 35, pp. 9564–9576, 2022.
- [297] S. Chen, Z. Jie, and L. Ma, “Llava-mole: Sparse mixture of lora experts for mitigating data conflicts in instruction finetuning mllms,” *arXiv preprint arXiv:2401.16160*, 2024.
- [298] K. Huang, X. Yin, H. Huang, and W. Gao, “Modality plug-and-play: Runtime modality adaptation in llm-driven autonomous mobile systems,” in *ACM MobiCom*, 2025.
- [299] X. Wang, B. Zhuang, and Q. Wu, “Modaverse: Efficiently transforming modalities with llms,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 26 606–26 616.
- [300] D. Zhang, S. Qi, X. Xiao, K. Chen, and X. Wang, “Merge then realign: Simple and effective modality-incremental continual learning for multimodal llms,” *arXiv preprint arXiv:2503.07663*, 2025.
- [301] O. Batur İnce, A. F. Martins, O. Mac Aodha, and E. M. Ponti, “Sample-efficient integration of new modalities into large language models,” *arXiv e-prints*, pp. arXiv–2509, 2025.
- [302] L. Jing, Y. Gao, Z. Wang, W. Lan, Y. Tang, W. Wang, K. Zhang, and Q. Guo, “Evomoe: Expert evolution in mixture of experts for multimodal large language models,” *arXiv preprint arXiv:2505.23830*, 2025.
- [303] L. Chen, H. Zhao, T. Liu, S. Bai, J. Lin, C. Zhou, and B. Chang, “An image is worth 1/2 tokens after layer 2: Plug-and-play inference acceleration for large vision-language models,” in *European Conference on Computer Vision*. Springer, 2024, pp. 19–35.
- [304] Y. Shang, M. Cai, B. Xu, Y. J. Lee, and Y. Yan, “Llava-prumerge: Adaptive token reduction for efficient large multimodal models,” *arXiv preprint arXiv:2403.15388*, 2024.
- [305] C. Yang, Y. Sui, J. Xiao, L. Huang, Y. Gong, C. Li, J. Yan, Y. Bai, P. Sadayappan, X. Hu *et al.*, “Topv: Compatible token pruning with inference time optimization for fast and low-memory multimodal vision language model,” in *Proceedings of the Computer Vision and Pattern Recognition Conference*, 2025, pp. 19 803–19 813.
- [306] S. Zhang, Q. Fang, Z. Yang, and Y. Feng, “Llava-mini: Efficient image and video large multimodal models with one vision token,” *arXiv preprint arXiv:2501.03895*, 2025.
- [307] L. Yao, L. Li, S. Ren, L. Wang, Y. Liu, X. Sun, and L. Hou, “Deco: Decoupling token compression from semantic abstraction in multimodal large language models,” *arXiv preprint arXiv:2405.20985*, 2024.
- [308] Y. Li, Y. Wu, J. Li, and S. Liu, “Accelerating transducers through adjacent token merging,” *arXiv preprint arXiv:2306.16009*, 2023.
- [309] X. Tang, J. Qiu, L. Xie, Y. Tian, J. Jiao, and Q. Ye, “Adaptive keyframe sampling for long video understanding,” in *Proceedings of the Computer Vision and Pattern Recognition Conference*, 2025, pp. 29 118–29 128.
- [310] B. Li, Y. Zhang, D. Guo, R. Zhang, F. Li, H. Zhang, K. Zhang, P. Zhang, Y. Li, Z. Liu *et al.*, “Llava-onevision: Easy visual task transfer,” *arXiv preprint arXiv:2408.03326*, 2024.
- [311] S. Bai, K. Chen, X. Liu, J. Wang, W. Ge, S. Song, K. Dang, P. Wang, S. Wang, J. Tang *et al.*, “Qwen2. 5-vl technical report,” *arXiv preprint arXiv:2502.13923*, 2025.
- [312] W. Pian, S. Deng, S. Mo, Y. Guo, and Y. Tian, “Modality-inconsistent continual learning of multimodal large language models,” *arXiv preprint arXiv:2412.13050*, 2024.
- [313] Z. Lin, M. Lin, L. Lin, and R. Ji, “Boosting multimodal large language models with visual tokens withdrawal for rapid inference,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 39, no. 5, 2025, pp. 5334–5342.
- [314] X. Shen, Y. Xiong, C. Zhao, L. Wu, J. Chen, C. Zhu, Z. Liu, F. Xiao, B. Varadarajan, F. Bordes *et al.*, “Longvu: Spatiotemporal adaptive compression for long video-language understanding,” *arXiv preprint arXiv:2410.17434*, 2024.
- [315] M. A. Baytas, D. Çay, Y. Zhang, M. Obaid, A. E. Yantaç, and M. Fjeld, “The design of social drones: A review of studies on autonomous flyers in inhabited environments,” in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, 2019, pp. 1–13.
- [316] Y. Liu, Z. Li, H. Liu, Z. Kan, and B. Xu, “Bioinspired embodiment for intelligent sensing and dexterity in fine manipulation: A survey,” *IEEE Transactions on Industrial Informatics*, vol. 16, no. 7, pp. 4308–4321, 2020.
- [317] A. Adilkhanov, M. Rubagotti, and Z. Kappassov, “Haptic devices: Wearability-based taxonomy and literature review,” *IEEE Access*, vol. 10, pp. 91 923–91 947, 2022.
- [318] Z. Wan, Y. Du, M. Ibrahim, J. Qian, J. Jabbour, Y. Zhao, T. Krishna, A. Raychowdhury, and V. J. Reddi, “Reca: Integrated acceleration for real-time and efficient cooperative embodied autonomous agents,” in *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, 2025, pp. 982–997.
- [319] Y. Li, L. Chen, A. Liu, K. Yu, and L. Wen, “Chatcite: Llm agent with human workflow guidance for comparative literature summary,” *arXiv preprint arXiv:2403.02574*, 2024.
- [320] V. Chheang, S. Sharmin, R. Márquez-Hernández, M. Patel, D. Rajasekaran, G. Caulfield, B. Kiafar, J. Li, P. Kullu, and R. L. Barra, “Towards anatomy education with generative ai-based virtual assistants in immersive virtual reality environments,” in *2024 IEEE international conference on artificial intelligence and eXtended and virtual reality (AIxVR)*. IEEE, 2024, pp. 21–30.
- [321] J. Wang, H. Xu, H. Jia, X. Zhang, M. Yan, W. Shen, J. Zhang, F. Huang, and J. Sang, “Mobile-agent-v2: Mobile device operation assistant with effective navigation via multi-agent collaboration,” *Advances in Neural Information Processing Systems*, vol. 37, pp. 2686–2710, 2024.
- [322] T.-K. Kuuru, “Embodied knowledge in customer experience: reflections on yoga,” *Consumption Markets & Culture*, vol. 25, no. 3, pp. 231–251, 2022.
- [323] Y. Liu, P. Li, C. Xie, X. Hu, X. Han, S. Zhang, H. Yang, and F. Wu, “Infigui-r1: Advancing multimodal gui agents from reactive actors to deliberative reasoners,” *arXiv preprint arXiv:2504.14239*, 2025.
- [324] Y. Xie, Z. Li, R. Shao, G. Chen, K. Zhou, Y. Li, D. Jiang, and L. Nie, “Mirage-1: Augmenting and updating gui agent with hierarchical multimodal skills,” *arXiv preprint arXiv:2506.10387*, 2025.
- [325] Y. Zhang, H. Jin, D. Meng, J. Wang, and J. Tan, “A comprehensive survey on process-oriented automatic text summarization with exploration of llm-based methods,” *arXiv preprint arXiv:2403.02901*, 2024.
- [326] J. Ni, T. Young, V. Pandelea, F. Xue, and E. Cambria, “Recent advances in deep learning based dialogue systems: A systematic survey,” *Artificial intelligence review*, vol. 56, no. 4, pp. 3055–3155, 2023.

- [327] M. Wu, J. Xu, and L. Wang, "Transagents: Build your translation company with language agents," in *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 2024, pp. 131–141.
- [328] Q. Yao, "Application of artificial intelligence virtual image technology in photography art creation under deep learning," *IEEE Access*, 2025.
- [329] J. Jiang, F. Wang, J. Shen, S. Kim, and S. Kim, "A survey on large language models for code generation," *arXiv preprint arXiv:2406.00515*, 2024.
- [330] C. Shivappa, "Audiobook applications: A revolution in reading literature," *New academia: An International Journal of English Language, Literature and Literary Theory*, no. April, pp. 24–31, 2023.
- [331] J. Piao, Y. Yan, J. Zhang, N. Li, J. Yan, X. Lan, Z. Lu, Z. Zheng, J. Y. Wang, D. Zhou *et al.*, "Agentsociety: Large-scale simulation of llm-driven generative agents advances understanding of human behaviors and society," *arXiv preprint arXiv:2502.08691*, 2025.
- [332] Z. Yin, Y. Yang, C. Hu, J. Li, B. Qin, and X. Yang, "Wearable respiratory sensors for health monitoring," *NPG Asia Materials*, vol. 16, no. 1, p. 8, 2024.
- [333] Y. Li, H. Wen, W. Wang, X. Li, Y. Yuan, G. Liu, J. Liu, W. Xu, X. Wang, Y. Sun *et al.*, "Personal llm agents: Insights and survey about the capability, efficiency and security," *arXiv preprint arXiv:2401.05459*, 2024.
- [334] G. He, G. Demartini, and U. Gadiraju, "Plan-then-execute: An empirical study of user trust and team performance when using llm agents as a daily assistant," in *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, 2025, pp. 1–22.
- [335] "Aloha 2: Github page," <https://aloha-2.github.io/>, accessed: 2025-09-22.
- [336] CVPR 2025 Workshop on Autonomous Driving, "Scaling up autonomous driving via large foundation models," <https://cvpr2025.wad.vision/>, 2025, accessed: 2025-09-18.
- [337] Amazon, "Amazon alexa+," <https://www.amazon.com/Meet-the-new-Alexa/dp/B0DCCNHWV5>, 2025, accessed: 2025-09-18.
- [338] G. R. Team, S. Abeyruwan, J. Ainslie, J.-B. Alayrac, M. G. Arenas, T. Armstrong, A. Balakrishna, R. Baruch, M. Bauza, M. Blokzijl *et al.*, "Gemini robotics: Bringing ai into the physical world," *arXiv preprint arXiv:2503.20020*, 2025.
- [339] ByteDance AI Lab, "Doubao: Bytedance large language model," <https://www.doubao.com/chat/>, 2025, byteDance's Doubao LLM series.
- [340] S. Zhang, S. Roller, N. Goyal, M. Artetxe, M. Chen, S. Chen, C. Dewan, M. Diab, X. Li, M. Lin *et al.*, "Opt: Open pre-trained transformer language models," *arXiv preprint arXiv:2205.01068*, 2022.
- [341] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo *et al.*, "Segment anything," pp. 4015–4026, 2023.
- [342] G. Team, R. Anil, S. Borgeaud, J.-B. Alayrac, J. Yu, R. Soricut, J. Schalkwyk, A. M. Dai, A. Hauth, K. Millican *et al.*, "Gemini: a family of highly capable multimodal models," *arXiv preprint arXiv:2312.11805*, 2023.
- [343] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *Journal of machine learning research*, vol. 21, no. 140, pp. 1–67, 2020.
- [344] Y. Li, S. Bubeck, R. Eldan, A. Del Giorno, S. Gunasekar, and Y. T. Lee, "Textbooks are all you need ii: phi-1.5 technical report," *arXiv preprint arXiv:2309.05463*, 2023.
- [345] S. Mukherjee, A. Mitra, G. Jawahar, S. Agarwal, H. Palangi, and A. Awadallah, "Orca: Progressive learning from complex explanation traces of gpt-4," *arXiv preprint arXiv:2306.02707*, 2023.
- [346] Z. Peng, W. Wang, L. Dong, Y. Hao, S. Huang, S. Ma, and F. Wei, "Kosmos-2: Grounding multimodal large language models to the world," *arXiv preprint arXiv:2306.14824*, 2023.
- [347] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Alten Schmidt, S. Altman, S. Anadkat *et al.*, "Gpt-4 technical report," *arXiv preprint arXiv:2303.08774*, 2023.
- [348] A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, and I. Sutskever, "Zero-shot text-to-image generation," pp. 8821–8831, 2021.
- [349] A. Srivastava, A. Rastogi, A. Rao, A. A. Shoeb, A. Abid, A. Fisch, A. R. Brown, A. Santoro, A. Gupta, A. Garriga-Alonso *et al.*, "Beyond the imitation game: Quantifying and extrapolating the capabilities of language models," *Transactions on machine learning research*, 2023.
- [350] P. Liang, R. Bommasani, T. Lee, D. Tsipras, D. Soylu, M. Yasunaga, Y. Zhang, D. Narayanan, Y. Wu, A. Kumar *et al.*, "Holistic evaluation of language models," *arXiv preprint arXiv:2211.09110*, 2022.
- [351] A. O'Neill, A. Rehman, A. Maddukuri, A. Gupta, A. Padalkar, A. Lee, A. Pooley, A. Gupta, A. Mandlekar, A. Jain *et al.*, "Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0," pp. 6892–6903, 2024.
- [352] C. Yeshwanth, Y.-C. Liu, M. Nießner, and A. Dai, "Scannet++: A high-fidelity dataset of 3d indoor scenes," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 12–22.
- [353] P. Lu, S. Mishra, T. Xia, L. Qiu, K.-W. Chang, S.-C. Zhu, O. Tafjord, P. Clark, and A. Kalyan, "Learn to explain: Multimodal reasoning via thought chains for science question answering," *Advances in Neural Information Processing Systems*, vol. 35, pp. 2507–2521, 2022.
- [354] Y. Liu, H. Duan, Y. Zhang, B. Li, S. Zhang, W. Zhao, Y. Yuan, J. Wang, C. He, Z. Liu *et al.*, "Mmbench: Is your multi-modal model an all-around player?" pp. 216–233, 2024.
- [355] B. Li, R. Wang, G. Wang, Y. Ge, Y. Ge, and Y. Shan, "Seed-bench: Benchmarking multimodal llms with generative comprehension," *arXiv preprint arXiv:2307.16125*, 2023.
- [356] C. Li, R. Zhang, J. Wong, C. Gokmen, S. Srivastava, R. Martín-Martín, C. Wang, G. Levine, M. Lingelbach, J. Sun *et al.*, "Behavior-1k: A benchmark for embodied ai with 1,000 everyday activities and realistic simulation," pp. 80–93, 2023.
- [357] L. Fan, G. Wang, Y. Jiang, A. Mandelkar, Y. Yang, H. Zhu, A. Tang, D.-A. Huang, Y. Zhu, and A. Anandkumar, "Minedojo: Building open-ended embodied agents with internet-scale knowledge," *Advances in Neural Information Processing Systems*, vol. 35, pp. 18343–18362, 2022.
- [358] O. Mees, L. Hermann, E. Rosete-Beas, and W. Burgard, "Calvin: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks," vol. 7, no. 3. IEEE, 2022, pp. 7327–7334.
- [359] G. Gerganov and contributors, "llama.cpp: Llm inference in c++," <https://github.com/ggml-org/llama.cpp>, 2025, supports AVX/NEON/AMX, Metal, CUDA, HIP, Vulkan, SYCL; CPU+GPU hybrid; 1.5–8-bit quantization.
- [360] Apple Inc., "Core ml," <https://developer.apple.com/machine-learning/core-ml/>, 2025, wWDC23 release highlights on-device LLM and diffusion model inference.
- [361] Google Research, "Tensorflow lite," <https://www.tensorflow.org/lite>, 2025, supports model quantization, pruning, and deployment of lightweight LLMs on mobile devices.
- [362] Alibaba Inc., "Mnn," <https://github.com/alibaba/MNN>, 2025, recent updates extend support for Transformer and LLM deployment on mobile/edge.
- [363] Tencent, "Ncnn," <https://github.com/Tencent/ncnn>, 2025, lightweight framework widely used on mobile; recent community releases include LLaMA and ChatGLM inference demos.
- [364] Huawei Technologies Co., Ltd., "Mindspore lite," <https://www.mindspore.cn/lite>, 2025, supports efficient deployment of Transformer and LLM models on Huawei Ascend NPUs and mobile devices.
- [365] Intel Corporation, "Openvino toolkit," <https://docs.openvino.ai>, 2025, latest release extends support for Transformer-based models including GPT, LLaMA, and BLOOM.
- [366] S. Gao, J. Yang, L. Chen, K. Chitta, Y. Qiu, A. Geiger, J. Zhang, and H. Li, "Vista: A generalizable driving world model with high fidelity and versatile controllability," *Advances in Neural Information Processing Systems*, vol. 37, pp. 91560–91596, 2024.
- [367] Z. Wu, X. Chen, Z. Pan, X. Liu, W. Liu, D. Dai, H. Gao, Y. Ma, C. Wu, B. Wang *et al.*, "Deepseek-vl2: Mixture-of-experts vision-language models for advanced multimodal understanding," *arXiv preprint arXiv:2412.10302*, 2024.
- [368] Z. Chen, X. Yang, J. Lin, C. Sun, K. Chang, and J. Huang, "Cascade speculative drafting for even faster llm inference," *Advances in Neural Information Processing Systems*, vol. 37, pp. 86226–86242, 2024.
- [369] H.-k. Chiu, R. Hachiuma, C.-Y. Wang, S. F. Smith, Y.-C. F. Wang, and M.-H. Chen, "V2v-llm: Vehicle-to-vehicle cooperative autonomous driving with multi-modal large language models," *arXiv preprint arXiv:2502.09980*, 2025.
- [370] K.-T. Tran, D. Dao, M.-D. Nguyen, Q.-V. Pham, B. O'Sullivan, and H. D. Nguyen, "Multi-agent collaboration mechanisms: A survey of llms," *arXiv preprint arXiv:2501.06322*, 2025.