REVIEW

National Science Open
5: 20260016, 2026
https://doi.org/10.1360/nso/20260016

Information Sciences

A survey on edge multimodal large models: Compression, inference acceleration, and applications

Siru Chen, Yuanchao Shu*, Cong Wang & Jiming Chen*

College of Control Science and Engineering, Zhejiang University, Hangzhou 310027, China

*Corresponding authors (emails: ycshu@zju.edu.cn (Yuanchao Shu); cjm@zju.edu.cn (Jiming Chen))

Received 31 January 2026; Revised 19 April 2026; Accepted 21 April 2026; Published online 29 April 2026

Abstract: Deploying multimodal large language models (MLLMs) at the network edge is critical for enabling low-latency, privacy-preserving multimodal intelligence. However, the substantial computational and memory demands of MLLMs present significant challenges for deployment on heterogeneous and resource-constrained edge devices. This survey systematically reviews existing approaches aimed at addressing these challenges. We categorize the literature along two complementary dimensions: model-level compression, which focuses on efficient architectural design and parameter reduction, and system-level inference acceleration, which emphasizes runtime optimizations such as scheduling and resource management. In addition, the survey examines the practical applications of edge-deployed MLLMs in domains such as cyber intelligence and embodied intelligence, and discusses emerging research directions, including edge-native model architectures, to further improve the trade-off between intelligence capability and resource efficiency.

Keywords: edge computing, multimodal large language models, model compression, inference optimization, cyber intelligence, embodied intelligence

INTRODUCTION

The continuous advancement of artificial intelligence (AI) has seen Transformer-based [1] architectures revolutionize natural language processing (NLP) and computer vision (CV). Large language models (LLMs), such as T5 [2], GPT series [3–5], and BERT [6], have achieved unprecedented success through massive pretraining and autoregressive generation. This foundation has paved the way for vision-language model (VLM) and multimodal fusion [7, 8], shifting the AI frontier toward multimodal large language models (MLLMs) that unify heterogeneous data types (e.g., text, images, video) into a single cohesive framework [9–12]. In the development of MLLMs, early models like CLIP [7] focused on cross-modal alignment, while later models such as Flamingo [13], PaLM-E [14], and BLIP-2 [15] enhanced reasoning capabilities, laying the groundwork for more complex multimodal understanding. Today, cutting-edge models like the GPT series [16] and Gemini series [17] are pushing MLLMs to new technological heights with their exceptional cross-modal perception and multi-turn interaction capabilities.

The transition of these powerful MLLMs from cloud-centric infrastructures to edge intelligence is driven by the escalating demand for data privacy and real-time responsiveness in autonomous driving, robotics, and unmanned aerial vehicles (UAVs) [18–20]. Edge devices are increasingly equipped with a variety of

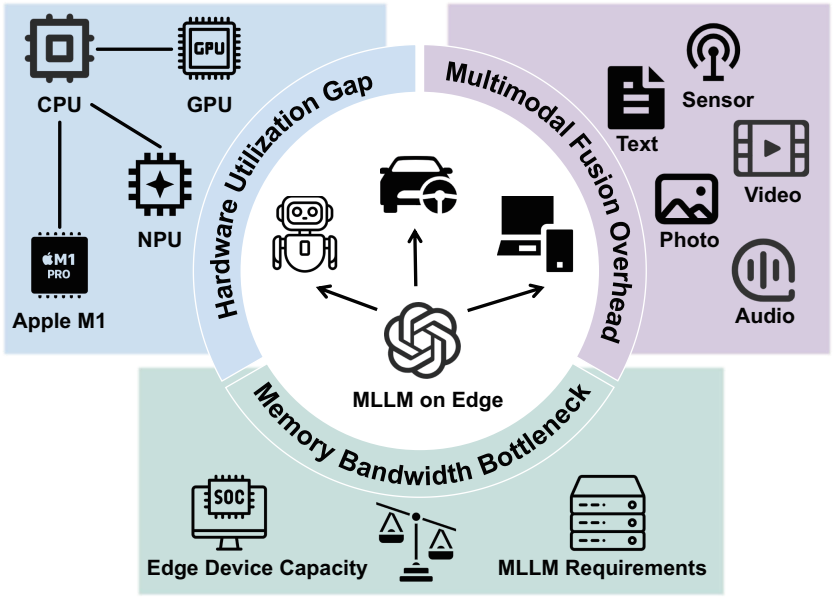

Figure 1 Challenges of MLLMs inference on edge.

sensors and some reasoning capabilities. This shift enables low-latency local execution and personalized experiences, while also reducing the risks associated with cloud offloading [21, 22].

In this work, edge devices are considered across three practical categories based on their resource constraints. First, resource-constrained devices (e.g., conventional sensors and microcontrollers) have extremely limited computation, memory, and energy budgets, making them incapable of running MLLMs. Second, intermediate devices (e.g., smartphones and embedded platforms) provide moderate computational resources and can support simplified or partially optimized MLLMs. Third, comparatively resource-rich edge nodes (e.g., in-vehicle computers and edge servers), although equipped with GPUs or dedicated accelerators, still operate under practical constraints in terms of memory capacity, power consumption, and real-time requirements.

Focusing on edge platforms capable of supporting MLLM deployment, namely intermediate devices and resource-rich edge nodes—we observe that, despite their differing capabilities, they share several fundamental constraints compared to cloud environments. To characterize these limitations, we identify three key bottlenecks in practical edge deployment, as illustrated in Figure 1. There are still three key challenges in the practical implementation process:

(i) Memory bandwidth bottleneck. While scaling laws [23] enhance MLLM performance, the memory wall constrains the parameter explosion. On edge SoCs, the limited memory bandwidth often fails to sustain the rapid weight-fetching required for autoregressive inference, making I/O overhead a more significant bottleneck than raw computation [24, 25]. This necessitates model compression to reduce memory footprints and data movement.

(ii) Multimodal fusion overhead. Processing high-resolution, multi-sensory data leads to a token-explosion in the attention mechanism. The computational cost of cross-modal alignment and synchronization scales poorly with input resolution, often exceeding the real-time latency budget of edge devices. This necessitates efficient modality encoders and connectors [14, 26].

(iii) Hardware utilization gap. A significant gap exists between theoretical FLOPs reduction and actual inference speed on heterogeneous hardware (e.g., NPUs). General optimizations often suffer from low hardware utilization due to fragmented memory access and unoptimized kernels. Closing this gap requires system inference acceleration to ensure deterministic performance for embodied intelligence [27–29].

To fill the gaps, this survey proposes a multi-layered framework that integrates model compression and system-level inference acceleration strategies, aimed at helping researchers choose appropriate solutions based on varying needs. As shown in Figure 2, the Section "Model compression" introduces model compression methods for modality encoders, connectors, and LLM backbones, the Section "System inference acceleration" discusses inference optimization through computation scheduling and collaborative execution, and the Section "Applications" showcases emerging applications in cyber and embodied intelligence.

This work distinguishes itself through three core perspectives.

(i) Integrated optimization-to-deployment pipeline. While existing surveys [30, 31] often treat model pruning and system deployment as isolated tasks, ours provide a unified analysis bridging structural optimization (the Section "Model compression") with system acceleration (the Section "System inference acceleration").

(ii) Expansion to high-dimensional multimodality. Moving beyond standard bimodal (vision-language) frameworks [32], ours evaluate emergent multi-modal architectures specifically designed to handle the high-dimensional data streams typical of complex edge environments.

(iii) Edge-specific constraint alignment. Unlike scenario-agnostic reviews [33] that overlook multimodal overhead [32], ours focuses on the intersection of multimodal complexity and resource-constrained execution, addressing the unique trade-offs required for edge efficiency.

BACKGROUND

This section will delve into the modular architecture of MLLMs and their development in edge computing, identifying the key bottlenecks faced during the inference stage, to lay a theoretical foundation for the deployment and inference of MLLMs on edge.

Architecture of MLLMs

As shown in Figure 3, MLLMs typically adopt a decoupled modular paradigm to facilitate the ingestion and alignment of heterogeneous data streams [11, 12, 34]. This architectural framework usually consists of three collaborating components: The modality encoder, as the perception layer, transforms the raw input into high-dimensional feature representations. To bridge the semantic gap between different data types, the cross-modal connector performs alignment and projection, ensuring compatibility between non-textual tokens and the language embedding space. The cognitive core of the system, the LLM backbone, then ingests these fused representations, conducts high-level reasoning, and generates multimodal responses. This modular design not only enhances system scalability but also provides discrete entry points for targeted optimizations, such as pruning by module, which are crucial for addressing the resource constraints of edge intelligence.

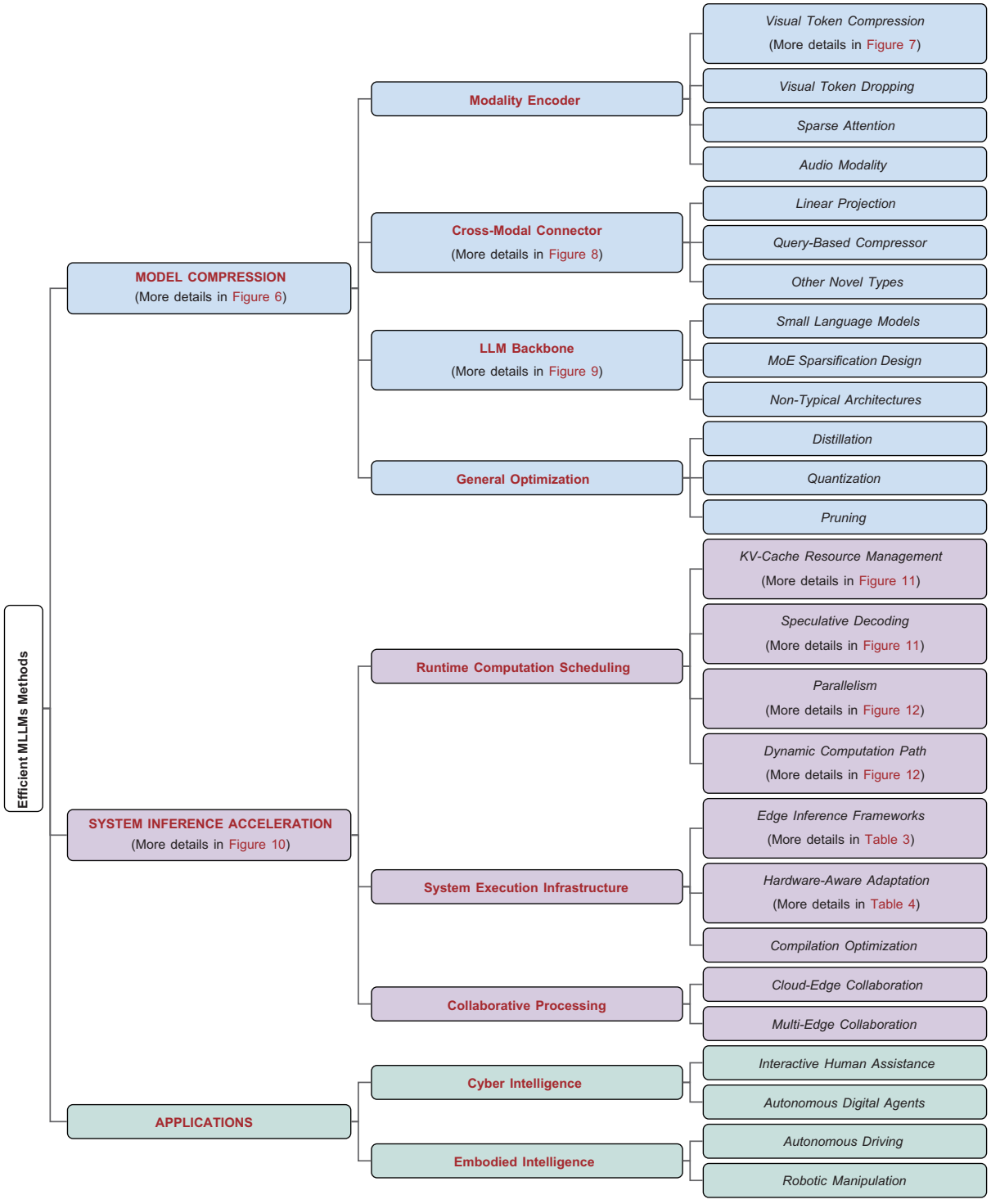

Figure 2 A comprehensive overview of MLLMs on edge.

Mainstream edge MLLMs

The development of edge MLLMs reflects a shift from modular optimization towards increasingly integrated architectures, driven by the need for real-time inference within the constrained power budgets of smart de-

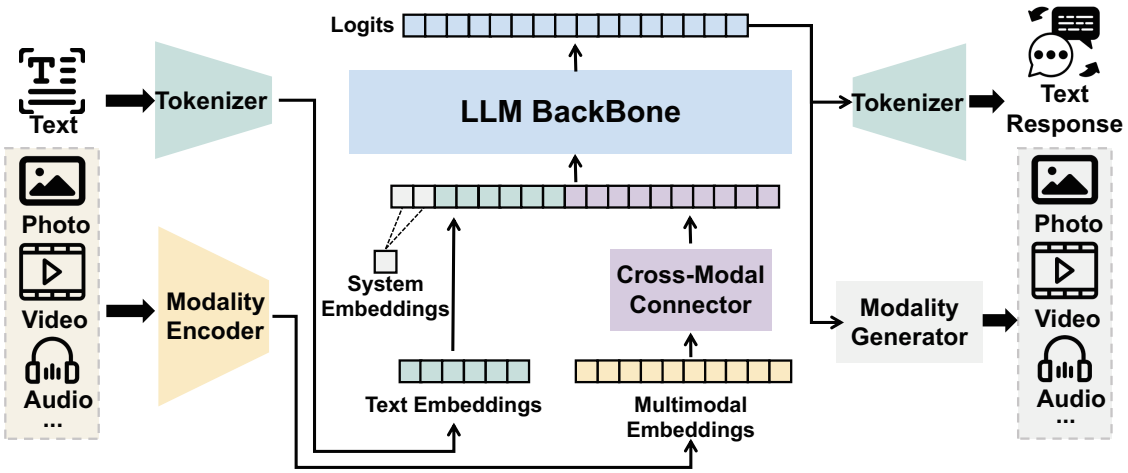

Figure 3 Typical architecture of MLLMs.

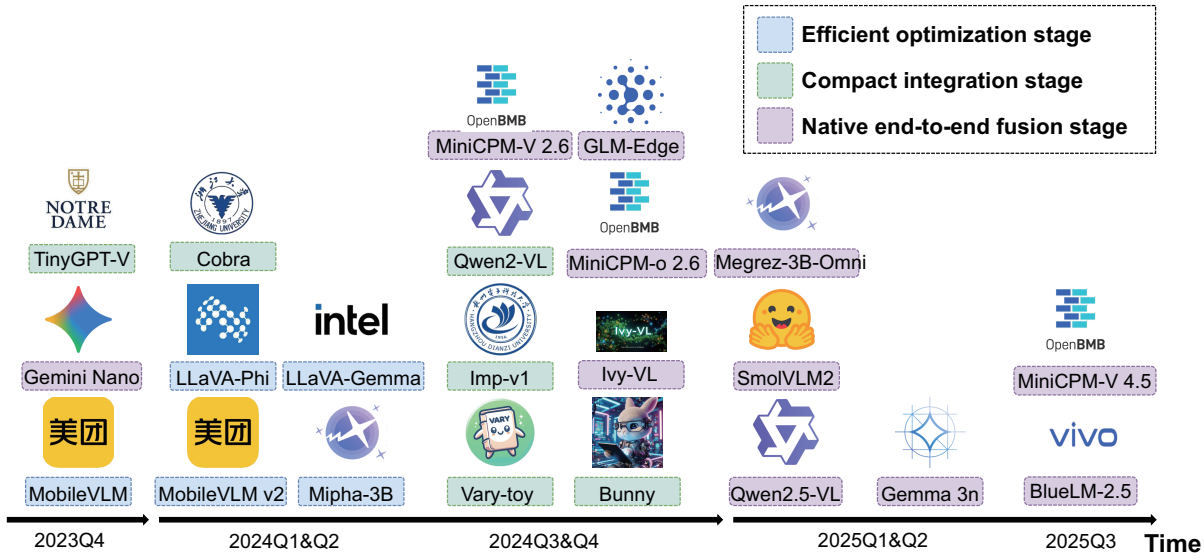

Figure 4 Timeline of end-to-end MLLMs on edge.

vices [35,36]. This evolution, as depicted in the timeline in Figure 4, can be conceptualized into three primary stages, each characterized by advancements in both structural integration and operational efficiency.

(i) Efficient optimization stage. The first phase marks the initial efforts to transition cloud-based intelligence to edge devices [37]. Early works such as BLIP-2 [15] and LLaVA [38] focused on reducing computational demands through techniques like pruning, distillation, and pooling. Notably, MobileVLM [39], LLaVA-Phi [40], and Mipha-3B [41] demonstrated the feasibility of deploying compressed MLLMs on mobile hardware while maintaining the core structure of the model. These efforts are primarily aimed at improving efficiency by optimizing computational resources, laying the foundation for more compact edge deployments.

(ii) Compact integration stage. In the second phase, the introduction of small language models (SLMs), such as Phi-2 [42] and TinyLlama [43], brought about further advancements in edge-based language gen-

Table 1 Bottleneck identification in the inference process across different tasks

Bottleneck location	Bottleneck description	Typical tasks	Related research
Modality encoder	Token explosion, high-dimensional feature heterogeneity	VQA, feature extraction, Video preprocessing	[51–54]
LLM backbone	Memory-intensive KV cache, massive parameter redundancy	Video captioning, language-guided generation	[10, 11, 55, 56]
Cross-modal connector	Alignment overhead, high memory-access ratio	Multimodal retrieval, content synthesis	[57–61]

eration and multimodal fusion. These models, being lightweight, were well-suited for processing natural language inputs and generating coherent responses within the resource constraints of edge devices. In this phase, researchers focused on the integration of language, vision, and audio modalities through cross-modal alignment layers and feature integration across encoders. The TinyGPT-V [44] and Vary-toy [45] models exemplified this stage, improving system efficiency through joint modality training, thereby optimizing both inference speed and semantic representation capabilities.

(iii) Native end-to-end fusion stage. The third phase represents a significant shift beyond the modular architectures of earlier stages, marked by native end-to-end integration of multimodal inputs. Models such as Gemini Nano [17] and Gemma 3n [46] leverage unified transformer frameworks to process heterogeneous data types—text, audio, and visual—through shared attention mechanisms and a common embedding space. This stage highlights the shift towards end-to-end multimodal fusion, where models are designed to handle multiple modalities seamlessly in a unified architecture. Further advancements can be observed in models like MiniCPM [47–49] and GLM-Edge [50], which demonstrate enhanced energy efficiency and multimodal processing capabilities, making them suitable for practical deployment in edge computing scenarios.

Bottlenecks in edge inference

The characteristic of edge MLLM inference is the presence of a multi-stage bottleneck effect, where hardware limitations amplify the computational inefficiencies of different modules. These bottlenecks are not a unified challenge but are highly dependent on the specific task and data. Table 1 [10, 11, 51–61] categorizes these key limitations, providing a roadmap for the subsequent discussion of optimization strategies.

Evaluation metrics and benchmarks

The deployment of edge MLLMs is a balancing act between physical constraints and cognitive fidelity. To provide a standardized selection guide for diverse application scenarios, we categorize evaluation into overhead metrics, functional utility benchmarks, and their inherent trade-off dynamics.

Overhead metrics

To quantify the cost of edge inference, we define metrics across temporal, energy, and spatial dimensions. Given an input prompt sequence and N generated output tokens, the total inference latency L is decomposed following the standard two-stage Transformer inference paradigm [62, 63]:

$$L = \mathrm{TTFT} + \mathrm{TPOT} \cdot (N - 1), \tag{1}$$

where time-to-first-token (TTFT) characterizes the latency of the compute-bound prefill stage, while time-per-output-token (TPOT) quantifies the incremental latency of the memory-bound autoregressive decoding stage. For battery-sensitive nodes, the energy per query (EPQ) is defined by the power integral $EPQ = \int_0^T P(t)\mathrm{d}t$ [64]. Furthermore, the memory footprint (M_{total}) is the critical hard-limit, calculated as

$$M_{total} \approx M_{weights} + \sigma \cdot (S + N) \cdot L \cdot H \cdot d_{head}, \tag{2}$$

where the second term represents the KV-Cache overhead (L transformer layers, H attention heads per layer, d_{head} per attention head). The scaling factor σ accounts for the storage of both key and value vectors and the floating-point precision (e.g., FP16/FP32), and this KV-Cache term essentially defines the maximum context window supported by the hardware [65, 66].

Functional utility benchmarks

To measure the intelligence retained after optimization, we monitor the accuracy drop (ΔAcc) and perplexity. We categorize utility into a three-tier hierarchy to guide model selection.

(i) General perception (tier-1). Evaluated via *MME* [67] and *MMBench* [68], focusing on basic visual-language alignment and OCR accuracy.

(ii) Cognitive reasoning (tier-2). Evaluated via *MMMU* [69] and *MathVista* [70], testing if the compressed core still possesses expert-level logic and mathematical grounding.

(iii) Temporal consistency (tier-3). Evaluated via *Video-MME* [71], crucial for agentic video analytics where long-term dependency is required.

Reporting these metrics alongside the compression ratio allows developers to assess the "Intelligence-per-Watt" of a specific deployment.

The ALP trilemma and selection logic

Edge MLLM deployment is fundamentally governed by the accuracy-latency-power (ALP) trilemma, a manifestation of the intrinsic tension between stochastic fidelity and deterministic resource bounds. This interdependence is formalized as a constrained multi-objective optimization (MOO) problem [72]:

$$\min_{\theta \in \Theta} \mathcal{F}(\theta) = [-\mathrm{Acc}(\theta), \mathrm{Lat}(\theta), \mathrm{Pwr}(\theta)]^\top \quad \text{s.t.} \quad M_{peak}(\theta) \leqslant M_{limit}, \tag{3}$$

where Θ represents the architectural and quantization search space. As illustrated in Figure 5, no single configuration can simultaneously minimize all objectives; instead, the deployment space defines a Pareto front of non-dominated solutions. The ALP trade-off triangle characterizes the MOO space. The selection logic is partitioned into three distinct regimes based on the application's utility function.

(i) Latency-first. For latency-sensitive applications like AR/VR or autonomous navigation, the objective function prioritizes TTFT and TPOT. This regime typically tolerates a performance degradation ($\Delta\mathrm{Acc} \in [5\%, 10\%]$) through aggressive W4A8 quantization or structural pruning to satisfy hard real-time constraints.

(ii) Accuracy-first. In high-stakes domains such as medical diagnostics or industrial defect detection, accuracy is the primary surrogate for safety. Developers often utilize FP16 or high-bit (e.g., INT8) weights, intentionally sacrificing throughput to ensure cognitive reliability.

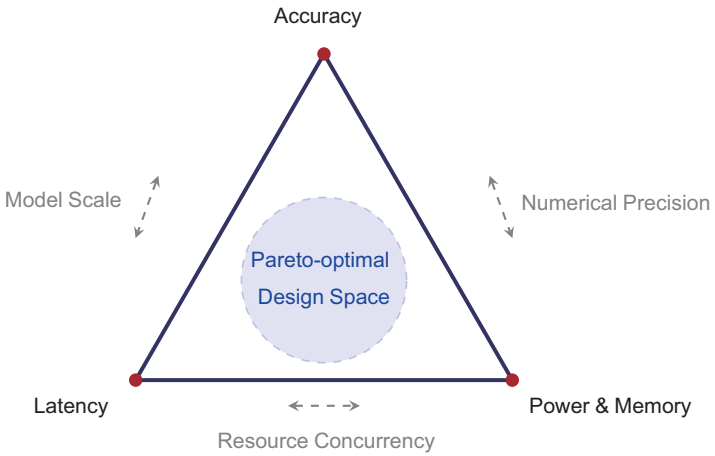

Figure 5 The ALP trade-off triangle. The edges represent the controllable variables—model scale, numerical precision, and resource concurrency—that drive the trade-offs between optimization objectives.

(iii) Energy-first. For battery-powered or "always-on" IoT sensors, EPQ is the critical bottleneck. This necessitates hardware-aware neural architecture search or dynamic voltage and frequency scaling to prolong the operational lifespan at the expense of peak throughput.

MODEL COMPRESSION

Model compression is crucial for deploying MLLMs on hardware with limited memory and power. Instead of adopting a single compression method, we classify existing methods according to the three MLLM architecture modules shown in Figure 3.

Specifically, as illustrated in Figure 6, the Section "Modal encoder" optimizations reduce input redundancy to save perception-level computation. the Section "Cross-modal connector" optimizations focus on efficient alignment to prevent latency bottlenecks between modalities. For the the Section "LLM backbone", methods like mixture of experts (MoE) or SLMs reduce decoding complexity. Finally, the Section "General optimization", such as quantization and pruning, provides a universal way to shrink the model footprint.

This modular classification approach helps developers select the most suitable compression tools based on specific edge computing constraints, while general optimization techniques offer flexible solutions for reducing the overall model size.

Modality encoder

Modality encoders function as the front-end perception layer, mapping heterogeneous raw inputs into unified semantic spaces. In edge scenarios, the visual encoder represents the primary computational bottleneck due to the massive token sequences generated from high-resolution inputs [73–75]. Consequently, current optimization research focuses on reducing visual token redundancy while preserving semantic integrity.

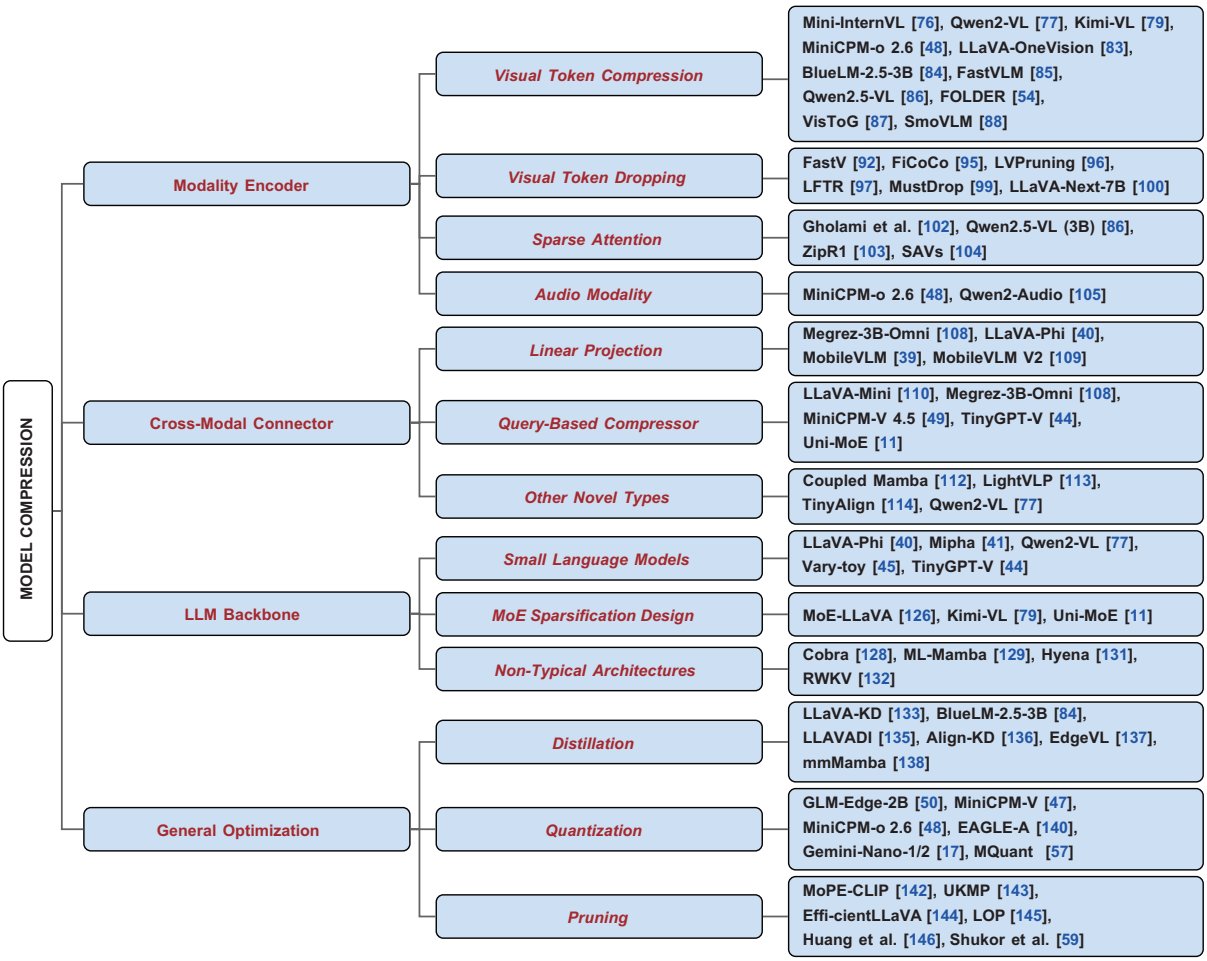

Figure 6 A comprehensive overview of model compression.

Visual token compression

Visual tokens capture fine-grained semantic details but exhibit unique redundancy patterns, such as spatial correlation and temporal continuity, which differ from text. This challenge intensifies with multi-image or ultra-high-resolution inputs, where visual tokens dominate inference costs and limit effective context length. To address this, as shown in Figure 7, compression methods are categorized into four approaches that balance computational savings with semantic integrity.

(i) Dynamic resolution and spatial downsampling. To manage the constraints of edge power, dynamic resolution processing adaptively scales input tokens to preserve essential visual details. For instance, Mini-InternVL [76] utilizes dynamic inputs to handle 4K autonomous driving data, while Qwen2-VL [77] introduces a naive dynamic resolution mechanism that eliminates absolute position embeddings via 2D-RoPE [78]. Other models like Kimi-VL [79] and MiniCPM-o 2.6 [48] further validate this by enabling edge devices to handle larger tokens through adaptive encoding like LLaVA-UHD [80], NaViT [81], and SigLIP [82]. In parallel, spatial downsampling reduces tokens by lowering resolution through pooling or convolutional layers. LLaVA-OneVision [83] utilizes bilinear interpolation and pooling for token compression, complemented by an AnyRes strategy to crop or downsample high-resolution inputs. Similarly, BlueLM-2.5-

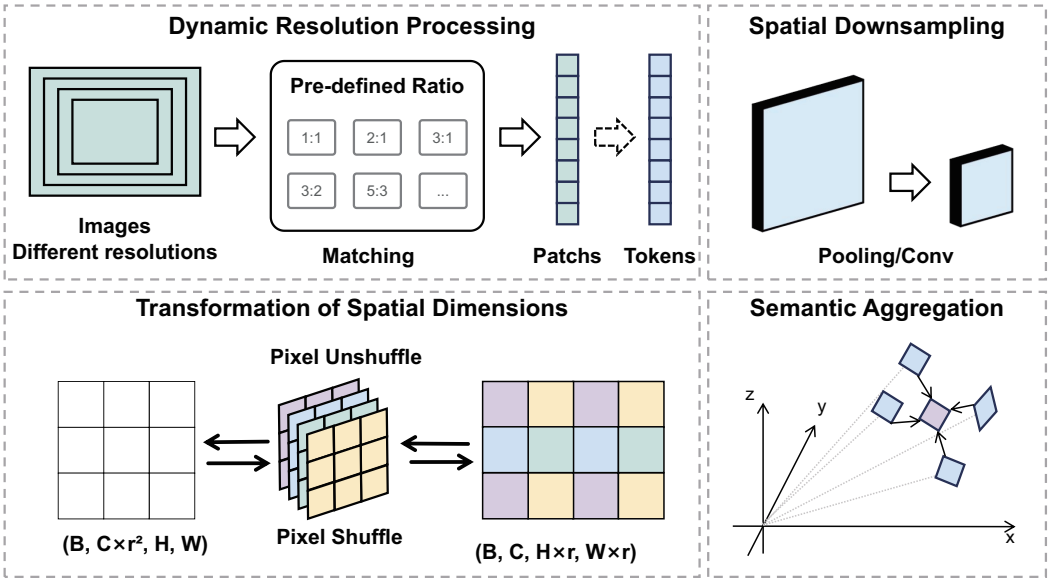

Figure 7 Methods for token reduction in the visual encoder of MLLMs.

3B [84] combines convolutional downsampling with dynamic resolution to minimize latency with negligible accuracy loss. FastVLM [85] introduces the FastViTHD structure, providing controllable token counts (e.g., 16 to 256 tokens) across various scales to bypass the limitations of traditional patch-splitting.

(ii) Semantic aggregation and dimensional transformation. In deeper stages of the encoder, semantic aggregation merges tokens based on feature similarity. Qwen2.5-VL [86] merges adjacent patches via multilayer perceptron (MLP) projection to preserve structure, while FOLDER [54] introduces a fold-and-merge module to cluster tokens by cosine similarity, achieving a 70% token reduction and 2× acceleration. Similarly, VisToG [87] uses learnable centers for semantic grouping to maintain high performance during compression. Furthermore, spatial-dimension dynamic transformation rearranges features to reduce sequence length for the downstream LLM. This is achieved either through direct compression using Pixel Unshuffle to map high-resolution features into fewer spatial positions, like Mini-InternVL [76], or progressive compression using Pixel Shuffle to layer-wise eliminate redundant computation. SmolVLM [88] utilizes the latter to achieve a compression ratio of $R = 4$, significantly enhancing model adaptability for edge deployment.

Architectural trade-offs. Although these methods effectively reduce token counts, a core trade-off exists between efficiency and fidelity. Dynamic resolution and semantic aggregation typically offer better performance but introduce non-deterministic computation paths and scheduling delays on real-time hardware. In contrast, static spatial downsampling and dimensional transformations provide stable inference latency and are more easily optimized for hardware like NPUs, making them ideal for latency-sensitive tasks.

Visual token dropping

High-resolution multimodal processing often generates significant redundancy in deep-layer tokens, leading to wasted computational cycles on edge [89, 90]. Unlike compression methods that modify features, token dropping selectively discards redundant units without retraining or weight modification. This is exemplified by Patch Slimming [91], which utilizes error backpropagation to identify and prune non-essential patches.

To optimize this process, current research focuses on three dimensions: attention distribution, modal synergy, and temporal continuity. FastV [92] exploits the attention variance between shallow and deep layers to achieve a 45% computational reduction in models like LLaVA-1.5 [93] and QwenVL-Chat [94]. For task-specific efficiency, FiCoCo [95] introduces spatially uniform dropping optimized by text-relatedness, yielding an 80% FLOPs reduction, while LVPruning [96] dynamically retains tokens based on linguistic context. In video scenarios, LFTR [97] within Video-LLaVA [98] merges adjacent frames and drops irrelevant tokens to handle temporal redundancy. Comprehensive frameworks like MustDrop [99] integrate local merging and semantic filtering across the encoding and decoding phases. In LLaVA-Next-7B [100], MustDrop maintains competitive performance on benchmarks like MMBench [68] and TextVQA [101] using only 11.1% of visual tokens, effectively reducing cache overhead by 88.9%.

Sparse attention

Sparse attention addresses the $O(N^2)$ complexity of traditional Transformers, which is a critical bottleneck for high-resolution edge inference. Theoretical analysis confirms that MLLM attention maps are inherently sparse, with visual tokens typically receiving less focus than textual counterparts [102]. To capitalize on this, Qwen2.5-VL [86] implements a window attention to reduce complexity to $O(n)$, enabling efficient processing of native-resolution images without distortion.

Advancements in active sparsity control, such as ZipR1 [103], utilize post-training reinforcement learning to reduce the inference token ratio from 80% to 25% on Qwen-series models [77, 86] with minimal accuracy loss. Furthermore, the sparse attention vectors (SAVs) method [104] identifies a discriminative subset of attention heads (under 5%) in generative MLLMs to enhance efficiency without fine-tuning.

Audio modality

While visual tokens represent the primary computational bottleneck, edge-based audio stream processing faces similar challenges regarding sequence length. Current optimization research focuses on temporal compression at the encoder stage.

To balance latency and power consumption, MiniCPM-o 2.6 [48] introduces a streaming design using one-second audio block segmentation and causal attention to maintain real-time inference. To further reduce overhead, Qwen2-Audio [105] optimizes the encoder by pooling adjacent acoustic features, effectively mimicking the spatial downsampling techniques used in visual encoders to prevent the LLM from being overwhelmed by high-rate audio frames.

A prevailing trend involves aligning audio encoder outputs with the visual token space, allowing edge hardware to reuse unified tensor acceleration operators. In contrast, lightweight encoding optimizations for other modalities, such as 3D point clouds or heterogeneous sensors, remain in the early stages of exploration, primarily due to data sparsity and the lack of standardized representation protocols for edge deployment.

Summary of modality encoder optimization

The optimization of modality encoders requires a strategic balance between compression depth and deployment stability. While token compression methods, such as spatial downsampling and pixel unshuffle, provide

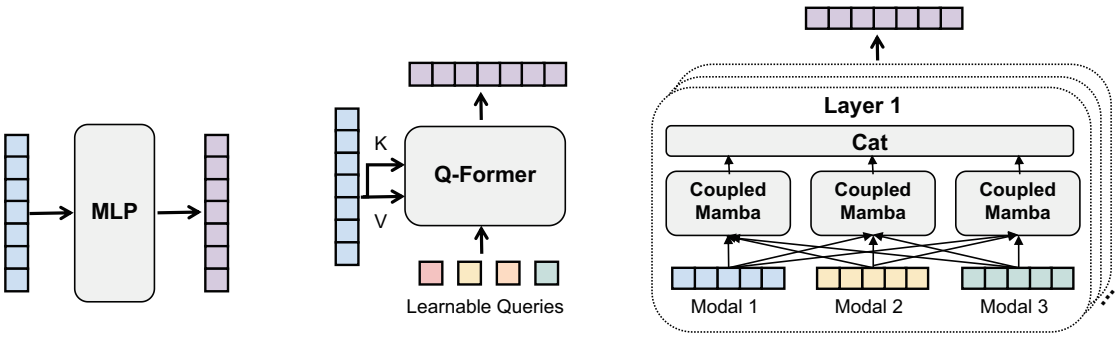

Figure 8 Optimized designs for cross-modal connectors in MLLMs. (i) Linear projection layer, (ii) query-based compressor, and (iii) example of novel connectors (e.g., Coupled Mamba).

predictable latency and high NPU compatibility, dynamic alternatives, like semantic aggregation, often introduce scheduling overhead. Similarly, token dropping techniques can achieve extreme reduction ratios, yet their practical acceleration is frequently limited by non-continuous memory access on edge hardware. Furthermore, although sparse attention effectively bypasses $O(N^2)$ complexity, its efficiency is highly dependent on the availability of optimized kernel operators. In practice, developers must weigh the high theoretical savings of dynamic strategies against the deterministic execution required for real-time edge applications.

Cross-modal connector

The cross-modal connector is the pivotal interface for aligning heterogeneous features, such as vision and speech, into the unified semantic space of the LLM. In edge computing environments, this module transcends simple modal bridging. It must serve as an information filter to prevent high-dimensional token explosion from overwhelming the downstream language backbone [106]. Key design priorities on edge include mitigating token count mismatches across modalities and minimizing the memory footprint associated with dense feature projections [11, 107]. Consequently, contemporary connectors are optimized not only for representational fidelity but also as secondary compression stages to ensure low-latency inference. As shown in Figure 8, current architectural designs for cross-modal alignment are categorized into linear projection, query-based compressors, and other novel structures.

Linear projection

Linear projection layers utilize single or MLPs to map heterogeneous features into the LLM embedding space. While structurally simple, these connectors offer negligible inference latency and a minimal parameter footprint, facilitating real-time execution on resource-constrained edge hardware. As demonstrated in the audio modality, Megrez-3B-Omni [108] employs a two-layer MLPs to direct Whisper-large-v3 features to the LLM, generating only 50 tokens per second with a trivial 0.003B parameter increase. This minimal overhead enables complex speech Q&A tasks within a 30 s window, proving the efficiency of linear alignment for edge speech processing. In the visual modality, the use of linear projectors focuses on balancing representational capacity with sequence compression. LLaVA-Phi [40] utilizes a two-layer MLP to maintain multimodal capabilities within a 3B-parameter framework while curbing computational power

consumption. To further address the token explosion problem, MobileVLM [39] introduces a lightweight downsampling projector (LDP) that leverages depthwise convolution to achieve a 75% reduction in visual tokens. MobileVLM V2 [109] iterates on this design with LDPv2 and conditional positional encoding, enhancing semantic alignment without compromising compression efficiency. These advancements illustrate that even with limited expressive capacity, optimized linear mappings remain a vital solution for low-latency edge deployment.

Query-based compressor

Query-based compressors utilize learnable query vectors to extract semantic essentials from dense token sets, effectively decoupling input resolution from the MLLM's computational budget. While foundational structures like Q-Former [15] demonstrate strong representational power, their slow convergence and spatial information loss have prompted the development of more efficient variants tailored for edge inference. As illustrated in Figure 8(ii), these methods are categorized by their compression intensity and architectural complexity.

(i) Aggressive and fixed-length compression. To meet the extreme constraints of mobile hardware, LLaVA-Mini [110] implements a compression module that condenses an entire image into a single visual token, minimizing memory footprint while preserving core semantic integrity. For tasks requiring higher perceptual fidelity, such as OCR, Megrez-3B-Omni [108] employs a Perceiver Resampler with 64 learnable queries. By applying cross-attention to 640 initial tokens from the SigLIP extractor, it achieves a 10× compression ratio with only 0.036B additional parameters. Similarly, MiniCPM-V 4.5 [49] utilizes cross-attention resampling to generate fixed-length sequences, ensuring stable inference latency regardless of input resolution.

(ii) Hybrid and multimodal strategies. To optimize parameter efficiency, TinyGPT-V [44] adopts a hybrid three-layer mapping that integrates a Q-Former with dual linear layers, reusing pre-trained components from BLIP-2 [15] and MiniGPT-4 [111] to minimize new training overhead. This paradigm extends beyond vision. Uni-MoE [11] introduces specialized Audio-Qformer and Speech-Qformer modules, distilling non-visual features into fixed-length queries to achieve unified linguistic alignment.

The selection of a query-based compressor hinges on the target application: extreme compression like LLaVA-Mini [110] is ideal for ultra-lightweight scenarios, whereas fixed-length resampling like Megrez-3B-Omni [108] provides the necessary precision for detailed visual reasoning. Collectively, these methods significantly alleviate resource pressure in edge environments by maintaining a compact yet semantically rich input space for the LLM backbone.

Other novel types

Beyond standard projections and compressors, novel cross-modal connectors leverage innovative architectures and functional enhancements to further optimize edge performance.

(i) Novel architectural shifts. To bypass the computational bottlenecks of Transformers, recent research has explored alternative sequence-modeling or hybrid structures. A representative example is Coupled Mamba [112], which, as illustrated in Figure 8(iii), replaces traditional fusion modules with a coupled state space model. This structure significantly enhances inference efficiency, delivering a 49% speedup and an

83.7% reduction in memory usage for 500-token sequences. Similarly, LightVLP [113] introduces a gated interaction mechanism, using dynamic gates to switch between cross-modal and unimodal representations, thereby improving robustness against noisy edge data.

(ii) Functional enhancements. Other methods focus on augmenting the expressive power of lightweight connectors. TinyAlign [114] integrates a RAG-Connector with a traditional two-layer MLP, utilizing a Perceiver structure to compress 100000 image-text pairs into just 5 tokens. This enhancement improves accuracy by 2.93% on the VQAv2 dataset [115] for the Phi-2 model [42] with minimal overhead. Furthermore, Qwen2-VL [77] introduces multimodal rotary position embedding (M-RoPE), which decomposes embeddings into temporal and spatial components. By unifying the alignment of text, image, and video, M-RoPE eliminates the need for additional modality conversions and reduces computational redundancy.

Summary of cross-modal connector optimization

The evolution of cross-modal connectors reflects a shift from simple linear mappings to structured, compression-aware architectures. **Linear projection layers** remain the benchmark for ultra-low latency in real-time tasks like speech, though they struggle with complex visual semantics. While **query-based compressors** offer high compression ratios, the associated cross-attention overhead can escalate pre-fill latency on resource-constrained devices. **Emerging novel structures**, such as Coupled Mamba [112], represent a promising trend by optimizing alignment efficiency for specific hardware instruction sets. Ultimately, the choice of connector must balance the required semantic granularity against the hardware's specific memory and computation constraints.

LLM backbone

The LLM backbone is the primary source of computational and memory overhead in MLLMs, and its efficiency determines the viability of edge deployment. To address these constraints, edge-specific architectural designs have evolved into three primary paradigms: (1) lightweight re-design of established large models to reduce structural redundancy; (2) training-from-scratch edge-first models that prioritize hardware-friendly primitives; and (3) adaptive or dynamic inference mechanisms that scale computational load based on input complexity. As illustrated in Figure 9, optimization strategies have transitioned from dense Transformer baselines to sparsified or non-typical architectures to balance inference speed with representational capacity.

Small language models

SLMs bridge the gap between lightweight re-design and training-from-scratch edge-first paradigms. While they often inherit proven architectural motifs from large-scale counterparts, they are typically optimized and trained from scratch with an explicit focus on 1.3–2.8B parameter scales to maximize on-device efficiency. Models such as InternLM [116], the Phi series [42, 117], TinyLlama [43], StableLM2 [118], MobileLLaMA [39], and Qwen [119] achieve performance parity with larger counterparts through optimized training and efficient multimodal interfaces. For example, LLaVA-Phi [40] and Mipha [41] utilize Phi-2-2.7B [42] to achieve results comparable to larger models like Pythia [120], particularly in image-driven code generation. Qwen2-VL [77] pairs a 2B backbone with a 675M ViT to reach an OCRBench [121] score of 809. Vary-

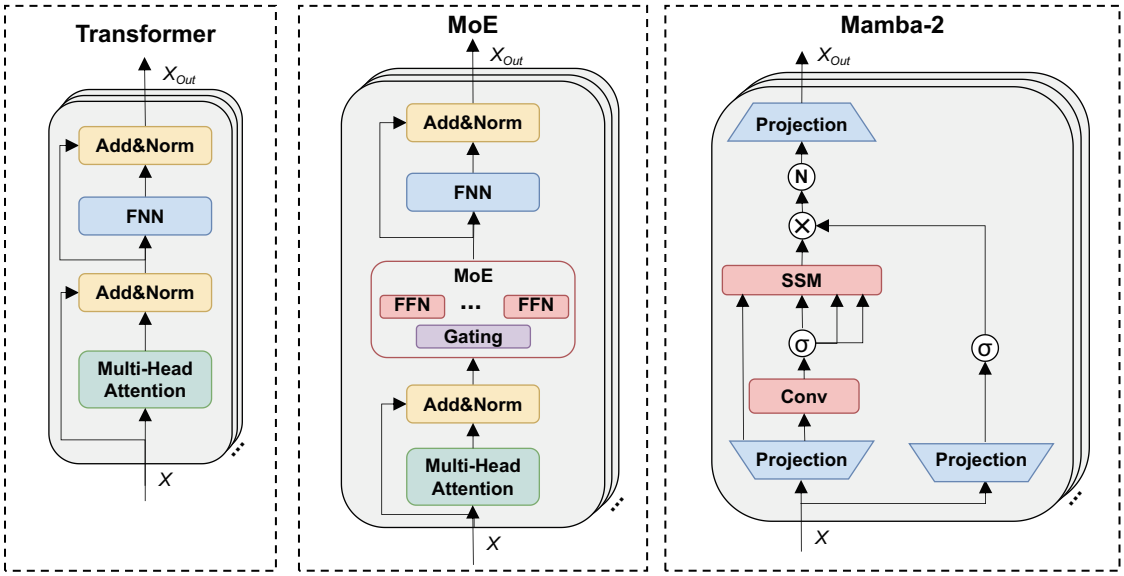

Figure 9 Comparison of transformer, MoE, and mamba architectures.

toy [45] employs Qwen-1.8B to reduce deployment costs on tasks like DocVQA [122] and RefCOCO [123]. To overcome training instability in small scales, TinyGPT-V [44] uses Phi-2-2.7B [42] and introduces input layer normalization, rotation magnitude suppression (RMS) Norm [124], and query-key normalization to mitigate gradient vanishing.

MoE sparsification design

The MoE architecture serves as a typical adaptive or dynamic inference mechanism, where inputs are routed through a sparse subset of experts to minimize per-token computation. To address the bottlenecks in feed-forward networks (FFN), MoE architectures [125] route inputs through a sparse subset of expert networks, reducing computational load without sacrificing total capacity. MoE-LLaVA [126] activates only a subset of FFN experts per token. Similarly, Kimi-VL [79] utilizes a sparse MoE approach to expand the language decoder to 16B parameters while activating only 2.8B during inference for significant speedup. Uni-MoE [11] combines shared self-attention with token-level sparse routing across multiple FFN experts, offering an efficient paradigm for adapting dense LLMs to sparse multimodal contexts. Future optimizations aim to further refine lightweight routing networks and expert pruning to minimize storage and routing overhead.

Non-typical architectures

Non-typical architectures, such as Mamba, represent a fundamental redesign of sequence modeling primitives to achieve linear complexity, offering a lightweight alternative to traditional Transformers. Some studies attempt to replace transformer backbone with non-typical architectures, fundamentally reconstructing the computational paradigm. A key example is the linear state space model (SSM), such as Mamba [127] and the S4 series. Unlike patch-based optimization methods such as convolution or sparse attention, SSM reduces the complexity of sequence modeling from quadratic to linear, offering new possibilities for edge multimodal inference. Recent research has successfully deployed SSM in edge MLLMs. Cobra [128] uses Mamba-

1 [127] as the LLM backbone, with a simplified vision-language alignment module. Cobra achieves a 4× speedup in inference using only 43% of LLaVA's [38] parameters, highlighting the potential of SSM-based multimodal inference. ML-Mamba [129] upgrades to Mamba-2 [130] and adds a vision-selective scanning connector, surpassing earlier Mamba models in both inference performance and task results. In contrast, other non-Transformer sequence models, like Hyena [131] and RWKV [132], show potential in edge multimodal applications but have not yet provided a complete deployment solution like Cobra, indicating that further exploration is needed.

Summary of backbone optimization

The choice of LLM backbone represents a trade-off between architectural maturity and theoretical efficiency. **SLMs** are the most mature path but may have depth limits in logical reasoning. **MoEs** reduce activation parameters but are often bottlenecked by weight-switching and memory bandwidth on edge devices. **Non-typical structures** like Mamba [127] address long-sequence complexity but currently face challenges with hardware ecosystem support and limited operator versatility.

General optimization

While module-specific optimizations provide targeted gains, general optimization strategies, including distillation, quantization, and pruning, offer a streamlined, cross-component approach to model lightweighting. These techniques focus on global parameter efficiency and bit-precision reduction, ensuring that the entire MLLM pipeline is compatible with the varying compute-and-memory envelopes of edge hardware.

Distillation

Knowledge distillation (KD) enhances the performance of student models by transferring high-level behavioral and feature-space knowledge from large-scale teachers. This approach is vital for enabling small-scale backbones to inherit the cross-modal alignment and reasoning capabilities of foundation models. Several studies highlight the effectiveness of distillation in edge MLLMs. LLaVA-KD [133] integrates MDist and RDist within a three-stage training framework to facilitate efficient cross-modal knowledge transfer without architectural modifications. BlueLM-2.5 [84] distills a 7B teacher into a 3B student, achieving performance comparable to Qwen3-4B [134] with only 60% of the parameters. LLAVADI [135] further optimizes this process using teacher-generated instructions to bridge the gap between 13 and 2.7B scales. Align-KD [136] specifically targets the student's first layer for cross-modal alignment distillation. For non-RGB modalities, EdgeVL [137] employs a dual-modal distillation method to transfer alignment from pretrained VLMs to compact models without manual annotation. mmMamba [138] introduces a three-stage progressive strategy to distill knowledge from dense transformers into linear-complexity Mamba-2 backbones, demonstrating the utility of KD in architectural shifts.

Quantization

Quantization serves as a core strategy for model-wide compression by mapping high-precision weights and activations to low-bit representations. While KV cache management (discussed in Section "System infer-

ence acceleration") targets dynamic inference memory, this section focuses on reducing the static memory bandwidth requirements of the entire MLLM pipeline. Modern edge MLLMs utilize hybrid and mixed-precision strategies to balance throughput with numerical stability. GLM-Edge-2B [50] achieves a decoding speed of over 60 tokens/s on the Snapdragon 8 Elite platform through hybrid quantization. Similarly, MiniCPM-V [47] and MiniCPM-o 2.6 [48] utilize GGML-based [139] 4-bit and GGUF formats to reach 3× compression ratios on mobile devices. To handle diverse data distributions, EAGLE-A [140] employs INT4/INT8 mixed-precision QAT to compress an 18 GB model to 3 GB for the iPhone 15 Pro. Gemini-Nano [17] further validates this by deploying INT4 models across varying memory tiers. The MQuant [57] framework addresses modality-specific distribution shifts using static quantization and RMS. Under W4A8 settings, MQuant reduces latency by 30% and memory usage by over 100% for models like InternVL2 [141] with minimal accuracy loss.

Pruning

Pruning reduces structural redundancy by eliminating non-essential weights or entire network units, focusing on maximizing FLOPs reduction while preserving multimodal fidelity. MoPE-CLIP [142] and UKMP [143] apply unified structured pruning to dual-tower and VLM architectures, respectively. For hardware-friendly execution, EfficientLLaVA [144] and LOP [145] use learning-driven schemes to predict optimal pruning strategies with minimal data overhead. Huang *et al.* [146] propose layer and width pruning strategies, highlighting width pruning's advantage in low-resource scenarios. To address redundancy during multimodal inference, Shukor and Cord [59] propose a static structural pruning strategy that skips transformer units at fixed intervals, preserving over 90% performance on VQA tasks.

Summary of general optimization

General optimizations provide the "last-mile" efficiency required for edge success. While distillation boosts accuracy and quantization improves throughput, their effectiveness is often bottlenecked by NPU instruction compatibility and the challenges of maintaining multimodal coordination during structural pruning. Current trends favor a hybrid approach, integrating quantization, pruning, and hardware-aware searches to adapt dynamically to diverse edge environments.

For a comprehensive engineering perspective, we provide a structured summary in Table 2 that compares representative techniques from this section across dimensions such as compression ratio, hardware assumptions, and modality support to facilitate practical deployment decisions.

SYSTEM INFERENCE ACCELERATION

While the model compression techniques discussed in the preceding section effectively mitigate computational overhead, they often overlook the dynamic execution behavior and resource contention intrinsic to edge devices. In practical deployments, structural reduction alone may not suffice to meet stringent real-time constraints due to the non-deterministic nature of resource availability.

As illustrated in Figure 10, the methodologies examined in this section optimize the end-to-end execution flow without altering the underlying model architecture. These strategies aim to achieve high-throughput,

Table 2 Comparative summary of MLLM compression techniques [a)]

Sec.	Technique	Core idea	Reduction type	Typical gain	Hardware
Modality Enc.	Visual Token Comp.	Resample / merge visual tokens before LLM	Token ↓	4x–16x tokens	GPU
	Token Pruning	Remove redundant tokens via saliency/attention	Token ↓	3x–18x tokens; 1.2x–1.8x latency	GPU
	Sparse Attention	Restrict attention patterns (local/global)	Complexity ↓	Moderate speedup	Custom kernels preferred
	Audio Modality	Downsample or compress temporal frames	Token ↓	2x–10x tokens	GPU
Cross-Modal Conn.	Linear Projection	Linear mapping for modality alignment	Token/Dim	~1x–4x	Universal
	Query-based	Bottleneck queries (e.g., Q-Former)	Token ↓	8x–20x tokens	Universal
	Other Connectors	pooling/routing variants	Token/Dim ↓	2x–10x	Universal
LLM Backbone	Small LMs	Replace backbone with smaller LLM	Params ↓	2x–10x params	Edge/GPU
	MoE	Activate subset of experts per token	Compute ↓	Throughput ↑	High bandwidth
	Non-typical	RNN / state-space architectures	Complexity ↓	Long-seq speedup	Kernel support helpful
General Opt.	Distillation	Transfer knowledge to smaller model	Params ↓	Variable	Universal
	Quantization	Low-bit weights (INT8/4, NF4)	Memory ↓	2x–4x memory	HW-dependent
	Pruning	Remove weights / channels	Params ↓	⩽2x	Sparse kernels

a) (1) "Reduction type" distinguishes token, parameter, memory, or computational complexity reduction. (2) Sparse attention, MoE, and state-space models are efficiency-oriented rather than strict compression. (3) Reported gains are component-level; end-to-end latency depends on system implementation.

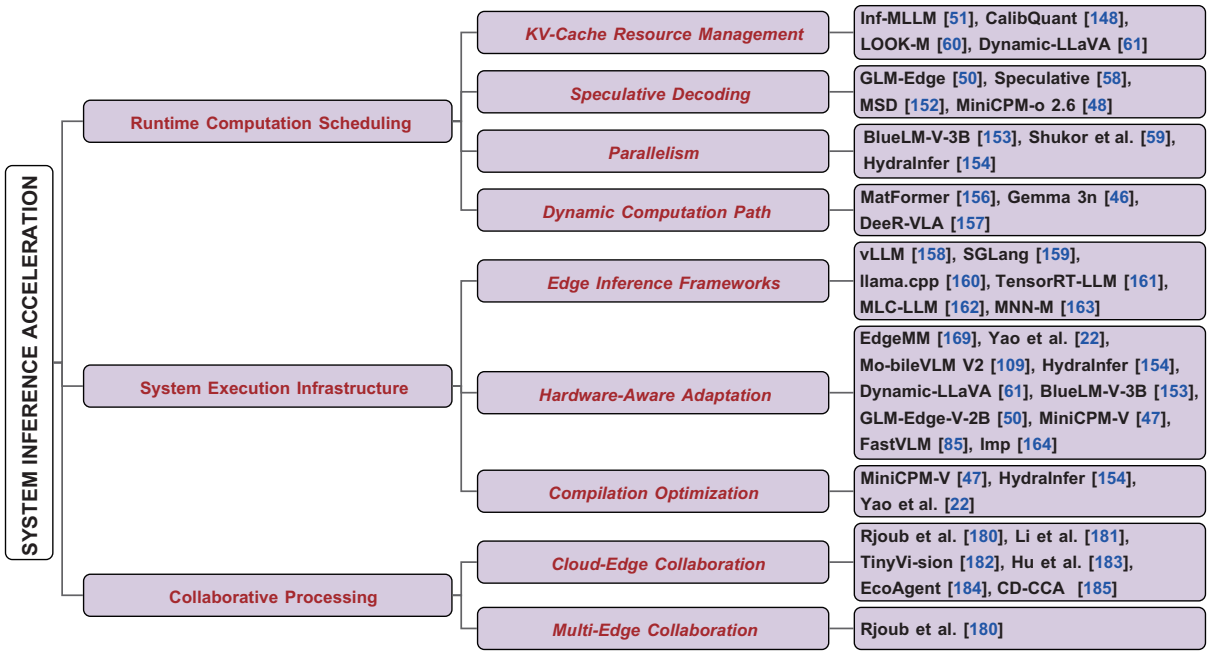

Figure 10 A comprehensive overview of system inference acceleration optimization.

low-latency, and energy-efficient multimodal inference by navigating the multi-dimensional constraints of computation, memory, and power. This is realized through a synergistic orchestration of runtime computation scheduling (the algorithmic logic), system execution infrastructure (the implementation stack), and collaborative processing (the distributed execution).

Runtime computation scheduling

Runtime computation scheduling optimization focuses on the strategic allocation of processing and memory resources to maximize inference throughput and minimize response latency on edge platforms. This subsection categorizes current advancements into four pivotal dimensions.

KV-cache resource management

The key-value (KV) Cache mechanism is fundamental to the efficiency of the transformer decoder in MLLMs. As depicted in Figure 11 (left), KV-Caching stores the key (K) and value (V) vectors to bypass redundant computations for historical tokens. However, the substantial memory footprint of MLLMs, exacerbated by high-resolution visual tokens, poses a significant challenge for memory-constrained edge nodes. Recent research has branched into three optimization trajectories.

(i) Sparsification. This approach targets the elimination of redundant or low-saliency tokens within the cache. For instance, Inf-MLLM [51] leverages the attention bias principle from Streaming-LLM [147] and introduces an attention saddle-point elimination mechanism. Maintaining a fixed-size KV-Cache, it facilitates long-context inference (e.g., 1-hour video) on single-GPU edge setups, significantly enhancing streaming efficiency.

(ii) Quantization. By transitioning from high-precision floating-point formats to low-bit representations, quantization reduces memory bandwidth requirements. CalibQuant [148] utilizes 1-bit quantization with post-scaling calibration, achieving a 10× throughput increase for the InternVL series [141] with negligible accuracy degradation when integrated with the Triton [149] runtime.

(iii) Multimodal scheduling. To handle the deluge of image tokens, LOOK-M [60] proposes a text-prioritized eviction strategy. By aggressively compressing image tokens while preserving text-token integrity, it reduces KV-Cache usage by 80%–95% and accelerates decoding by up to 1.5× on LLaVA [38] and MobileVLM [39]. Similarly, Dynamic-LLaVA [61] adaptively prunes the prefill computation and decoding memory, tailoring the sparsification intensity to the multimodal scenario.

Speculative decoding

The autoregressive nature of MLLM generation inherently introduces serial computation bottlenecks. As depicted in Figure 11 (right), Speculative decoding circumvents this by employing a lightweight draft model to hypothesize multiple tokens, which are subsequently verified in parallel by the target MLLM [150, 151]. Edge-specific implementations, such as GLM-Edge-1.5B [50] on the Snapdragon 8 Elite, demonstrate that parallelizing verification can drastically reduce the serial overhead of large-scale models. Gagrani *et al.* [58] achieved a 2.37× speedup for LLaVA-7B [38] by integrating a 115M draft model without modifying the visual encoder. Furthermore, MSD [152] addresses the multimodal bottleneck through text-visual token

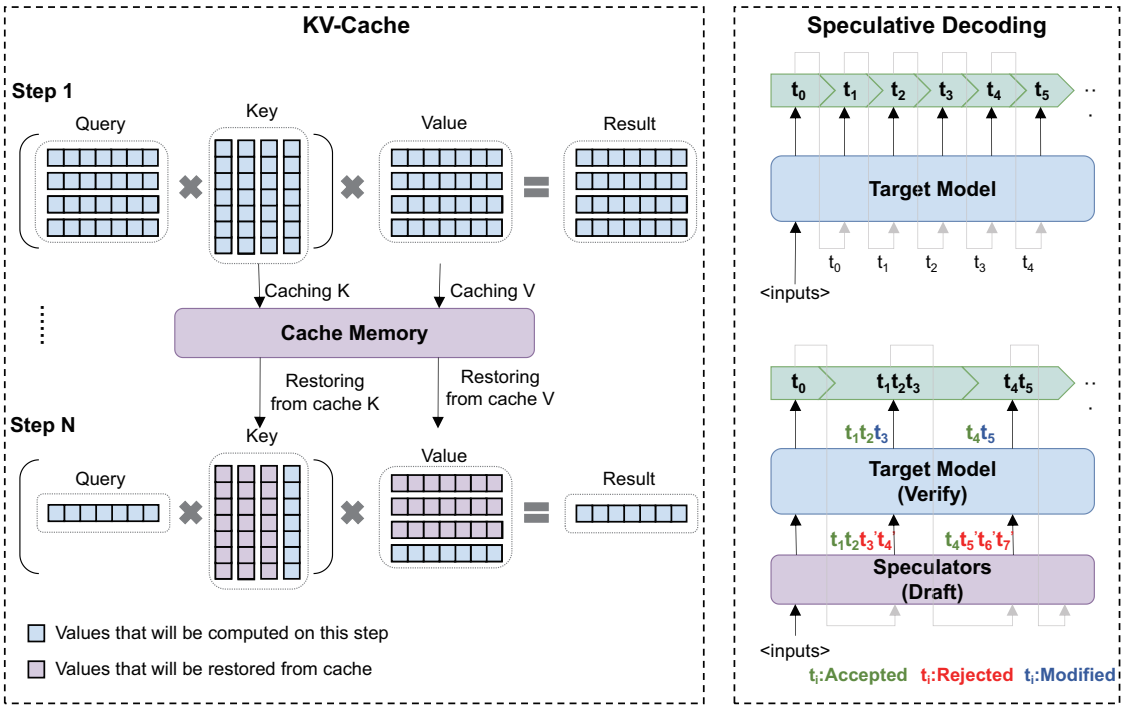

Figure 11 Comparison of *KV*-cache and speculative inference optimization strategies.

decoupling, yielding a 2.29× acceleration for LLaVA-1.5 [93]. In the audio domain, MiniCPM-o 2.6 [48] utilizes streaming decoding with causal attention masks to enable real-time, incremental speech-text interaction.

Parallelism

Pipeline parallelism partitions the model's depth into stages executed across heterogeneous hardware units, as shown in Figure 12 (right). The BlueLM-V-3B model [153] exemplifies this by offloading convolutional tasks to the CPU while concurrently executing ViT layers on the NPU. This staggered execution minimizes resource idling. Beyond structural parallelism, Shukor and Cord [59] propose transforming serial FFN and self-attention (SA) layers into parallel execution paths. HydraInfer [154] utilizes CUDA streams to achieve kernel-level parallelism between visual encoding and language processing, enhancing GPU throughput by 4× through instance-level collaboration.

Dynamic computation path

Dynamic computation path techniques encompass a broad spectrum of adaptive inference strategies, including layer skipping, early exiting, and variable attention mechanisms. To illustrate this category, Figure 12 (left) presents early exiting as a representative example. By adjusting the computation path based on input complexity and contextual redundancy, these methods enable on-demand inference, reducing both latency and energy consumption while preserving performance [59, 155]. A notable example within the broader context of dynamic paths is the MatFormer elastic architecture [156], which splits large models into smaller nested sub-models to achieve adaptive acceleration. Gemma 3n [46] leverages this nested structure to activate

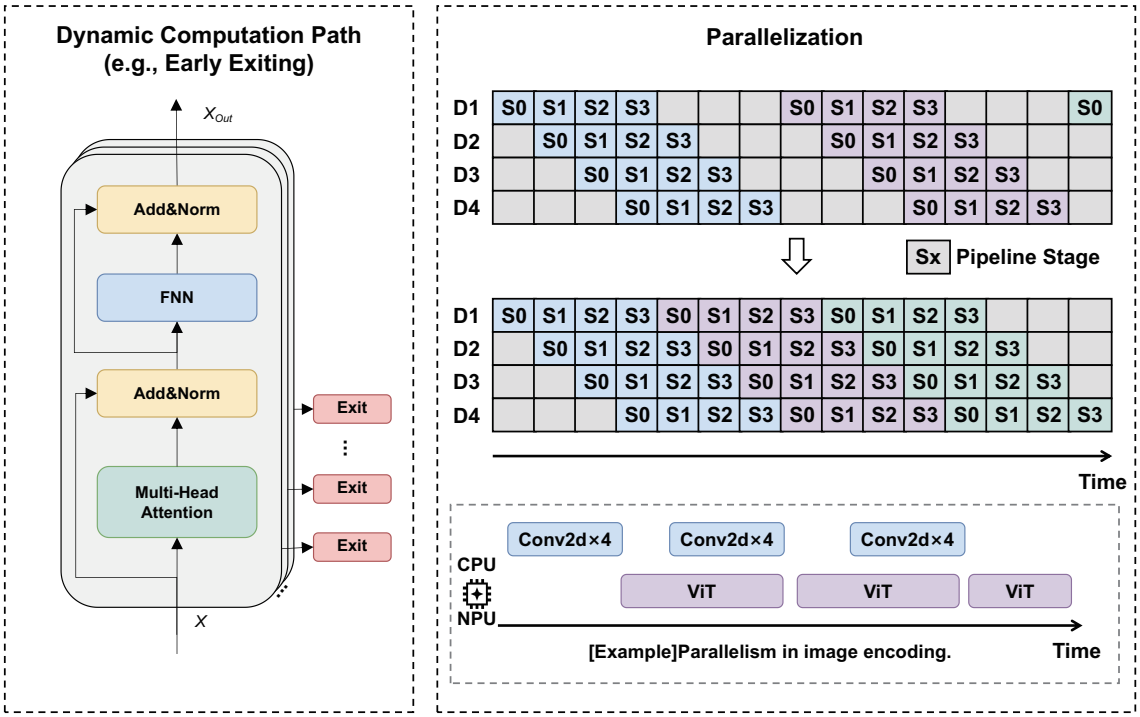

Figure 12 Comparison of parallelism optimization and dynamic computation path inference strategies. The left panel uses early exiting as a representative example to illustrate the concept of dynamic computation paths.

sub-models of varying depths dynamically. In robotics, where real-time response is critical, DeeR-VLA [157] integrates intermediate exits within the vision-language-action (VLA) models pipeline. By terminating inference once action consistency is reached, it significantly curtails computational costs in simple environments without compromising task success rates.

Summary of runtime computation scheduling

Runtime computation scheduling optimizations offer diverse pathways for inference acceleration without necessitating architectural re-training. While KV-Cache optimization like LOOK-M [60] effectively mitigates memory pressure, it may introduce irregular memory access patterns that impact cache-line efficiency. Speculative decoding like MSD [152] offers substantial speedups but remains sensitive to the alignment between the draft and target models. In cases of low-quality draft predictions, the resulting validation rollbacks may negate the benefits of parallelism. Finally, dynamic paths like MatFormer [156] provide adaptability. Yet, the overhead of the decision-making modules must be carefully balanced against the computational savings to ensure a net gain in edge efficiency.

System execution infrastructure

In edge MLLM inference, relying solely on software or hardware optimizations often cannot achieve both low latency and high throughput. Therefore, system execution infrastructure has become a key research focus. By integrating edge inference frameworks, hardware adaptation, and compilation optimization, both execution efficiency and energy performance can be significantly enhanced while maintaining model accuracy.

Table 3 Comparison of representative inference framework for MLLMs deployment [a)]

Framework	Core optimization	Key MLLM support	Multi-modal	Primary use case	Target platforms
vLLM [158]	PagedAttention, throughput	HF Hub seamless integration	✓	High-throughput batch processing / Stateless APIs	NV, AMD, TPU, AWS
SGLang [159]	RadixAttention, graph opt	Llama, DeepSeek, LLaVA	✓	Latent-sensitive chat / Complex agentic systems	NV, AMD GPUs
llama.cpp [160]	GGUF quantization, lightweight kernels	LLaMA series, Falcon, Qwen2-VL, LLaVA	✓	CPU-heavy inference / local prototyping	Apple, ARM, x86, NV, Mobile GPU
TensorRT-LLM [161]	Deep operator fusion, speculative decoding	GPT, MoE, quantized LLMs	–	Industrial-grade production on NVIDIA clusters	NV GPUs, Clusters
MLC-LLM [162]	TVM compiler-based, universal deployment	Llama, Mistral, GPT-2, Phi-3.5, Qwen3	–	Universal cross-platform "write-once-run-anywhere"	iOS, Android, ARM, NV/AMD/Intel
MNN [163]	Heterogeneous computing, mobile-specific kernels	Qwen, Yi, InternLM, SmolM, MiMo	✓	Ultra-resource-constrained mobile NPU/GPU tasks	Mobile NPU/GPU, Vulkan, Metal

a) NV: NVIDIA; HF Hub: Hugging Face Model Hub; GGUF: GPT-Generated Unified Format; TVM: Tensor Virtual Machine; NPU: Neural Processing Unit; AWS: Amazon Web Services; TPU: Tensor Processing Unit.

Edge inference frameworks

Efficient inference frameworks serve as the cohesive execution layer that integrates hardware-level optimizations with the dynamic computation scheduling strategies (e.g., KV-cache management and speculative decoding) discussed in the previous subsection. As summarized in Table 3 [158–163], contemporary frameworks bridge the gap between complex MLLM architectures and resource-constrained edge environments by orchestrating specialized backends and memory allocation policies.

Implementation of scheduling and memory logic. Rather than mere porting tools, modern edge frameworks are designed to operationalize advanced scheduling. For instance, lightweight edge-native frameworks like llama.cpp [160] and MLC-LLM [162] focus on minimizing the memory footprint and maximizing CPU-GPU co-inference. llama.cpp utilizes the GGUF format and optimized C/C++ kernels to bypass the heavy Python runtime. For instance, MobileVLM [39] leverages this framework on Jetson Orin to achieve a low latency of 65 ms, demonstrating that efficient visual token pre-filling (occupying $< 15\%$ of total time) is achievable via native optimization. MLC-LLM employs the TVM compiler stack to enable universal deployment. A hybrid approach in Imp [164] integrates both llama.cpp and MLC-LLM, achieving 13 tokens/s on Android smartphones. This highlights the effectiveness of modular time-breakdown optimization, specifically reducing prompt encoding and generation phases to millisecond scales.

Dynamic execution and high-throughput engines. To support more complex scheduling like PagedAt-

Table 4 Comparison of hardware architectures and adaptation strategies for edge MLLMs

Hardware	Key strengths	Critical constraints	Optimization focus	Related work
CPU	Serial logic, complex control	Low arithmetic intensity	SIMD/Vector acceleration	[166, 167]
GPU	Massive parallelism, throughput	Memory bandwidth wall	Sparse compute, memory reuse	[109, 154, 167]
NPU	Energy efficiency, low precision	Hard-coded kernels, latency	Quantization, Op offloading	[50, 153, 168]
SoC	Heterogeneous synergy	Inter-module data transfer	Adaptive task scheduling	[164]

tention and sparse execution, frameworks originally tailored for cloud-scale inference, such as vLLM [158] and SGLang [159], are increasingly adapted for edge gateways and high-end embedded systems. Through PagedAttention and dynamic *KV*-Cache management, Megrez-3B-Omni [108] combines vLLM with quantization to strike a balance between throughput and accuracy. Similarly, MiniCPM-o 2.6 [48] demonstrates the feasibility of rapid local prototyping by integrating vLLM with web-based user interface (UI) toolchains [165], ensuring compatibility with standard edge deployment pipelines.

To summarize, the trend in edge MLLM inference is shifting from static model porting to algorithm-aware system integration. By decoupling model logic from hardware execution, these frameworks significantly enhance cross-platform adaptability while maintaining the low-latency requirements of real-time edge applications.

Hardware-aware adaptation

The inherent heterogeneity of edge devices, ranging from general-purpose CPUs to dedicated NPUs, necessitates hardware-specific adaptation to bridge the gap between heavy MLLM workloads and constrained resources. Table 4 [50, 109, 153, 154, 164, 166–168] summarizes the characteristics and optimization focuses for different hardware backends.

(i) CPU & GPU: maximizing throughput. To address the dual challenges of limited CPU floating-point performance and GPU memory bandwidth bottlenecks in mobile heterogeneous computing, recent studies have focused on architectural co-design and parallel efficiency. For CPU-side enhancements, EdgeMM [169] integrates a systolic array co-processor to accelerate matrix operations, while Yao *et al.* [22] utilized automated parameter searching to dynamically match CPU core allocation with real-time decoding demands. Regarding GPU throughput, MobileVLM V2 [109] and HydraInfer [154] improve data exchange efficiency via customized CUDA kernels and optimized migration protocols. Meanwhile, Dynamic-LLaVA [61] employs batch-parallel sparse inference to maximize the saturation of GPU compute units. In mobile environments lacking vendor-specific frameworks, researchers have turned to cross-platform solutions like OpenCL [170]. For instance, Hyperion [171] maximizes GPU potential through optimized task offloading and scheduling. Furthermore, CoDL [172] explores CPU-GPU co-execution strategies to surpass the performance limits of individual hardware components through joint computation.

(ii) NPU: low-power specialization. As dedicated AI accelerators, NPUs are essential for energy-efficient MLLM inference on mobile devices. Current research achieves performance leaps through fine-grained hardware-software co-design. At the task allocation level, BlueLM-V-3B [153] and GLM-Edge-V-2B [50] offload compute-intensive backbones to the NPU to leverage its dedicated circuits for speed and power ef-

ficiency. Regarding execution efficiency, MiniCPM-V 4.5 [49] utilizes INT4 low-precision quantization to align with the NPU's fixed-point computing strengths, significantly increasing operational density per clock cycle. Similarly, FastVLM [85] attains a 6.9× acceleration by directly invoking native engines (e.g., Apple Neural Engine [173]) to map vision tasks into hardware-native instructions. Despite these gains, NPU performance is tightly coupled with architectural specialization, posing significant challenges in complex edge scenarios. Firstly, operator incompatibility remains a hurdle, as native libraries often lack support for non-standard or emerging model structures. Secondly, on-chip memory constraints—specifically the limited SRAM capacity restricted by silicon area and power budgets—hinder the caching of large-scale intermediate features. Finally, NPUs face throughput scalability issues under high-concurrency or large-batch workloads due to insufficient memory bandwidth and parallel execution scale, which restricts their versatility in multi-tasking environments [174].

(iii) SoC: unified resource management. System-on-Chip (SoC) designs shift the focus from single-unit optimization to cross-module co-execution. By integrating CPU, GPU, and NPU into a unified scheduling framework, SoCs can adaptively allocate tasks across different inference stages, typically assigning backbone inference to the NPU while utilizing the GPU for parallel pre-processing. A representative example is Imp [164], which implements a collaborative scheme on Snapdragon [175] to achieve seamless resolution adaptation and hardware-level task matching.

Crucially, the efficacy of such system-level resource management is fundamentally governed by the hardware-algorithm affinity (HAA), the structural and operational compatibility between compressed model motifs and hardware execution primitives. To bridge the gap between the Section "Model compression" and the Section "System inference acceleration", we identify three synergistic design principles as shown in Table 5.

(i) Operational intensity reconfiguration via quantization: Quantization, quantization strategies don't merely reduce storage; they reconfigure the hardware's roofline boundary. For memory-bound LLM decoding, transitioning from FP16 to INT4 doubles the operational intensity, allowing runtime schedulers to bypass the SoC's unified memory bandwidth wall. This bandwidth relief enables more aggressive KV-cache prefetching and larger batching, effectively shifting the execution bottleneck from I/O-bound to compute-bound and maximizing the utilization of NPU/GPU tensor cores [176, 177].

(ii) Execution pattern alignment via pruning: The compatibility between pruning paradigms and heterogeneous backends determines the actual speedup. While unstructured sparsity achieves high theoretical compression, it leads to SIMD execution stalls on mobile NPUs due to irregular memory indexing. In contrast, Structured Pruning aligns with the tile-based dispatching of edge accelerators. This affinity dictates a mapping rule: structured pruned sub-networks are statically offloaded to NPUs, while unpruned dense components are dynamically scheduled to the mobile GPU to leverage its higher clock frequency [178].

(iii) Modality-specific pipeline disentanglement: For MLLM workloads, a "divide-and-conquer" pipeline is optimized based on modality-specific tolerance. The visual encoder—highly amenable to Token Pruning and Visual Token Compression is offloaded to the NPU for low-power streaming. Concurrently, the LLM backbone utilizes mixed-precision quantization on the GPU to balance reasoning flexibility and throughput [179].

This HAA-aware mapping, where compression techniques act as deterministic inputs for system-level scheduling heuristics, transforms theoretical sparsity into tangible hardware-level performance gains.

Table 5 Cross-layer affinity matrix: Mapping compression to edge hardware dispatching

Compression paradigm	Hardware affinity	System-level optimization	Primary metric impact
Quantization	NPU/GPU	Operational intensity teconfiguration	↑ Throughput (batching)
Structured pruning	NPU (SIMD)	Execution pattern alignment	↓ Latency (vectorization)
Token pruning	NPU/DSP	SRAM footprint reduction	↓ Memory usage (KV-cache)
Mixed-precision MoE	SoC	Dynamic resource offloading	↓ Energy per inference

Compilation optimization

In addition to hardware adaptation, compilation optimization is essential for efficient model execution on target devices. This involves integrating the computation graph with the hardware architecture through model compilation and instruction set adaptation to improve execution efficiency. For instance, MiniCPM-V 4.5 [47] demonstrates that compilation-level graph optimization, by mapping operations to hardware-native instructions, effectively resolves compatibility issues and achieves near-custom hardware efficiency. HydraInfer [154] supports scheduling strategies by designing a unified fusion kernel that enables *KV*-Cache and image cache operations to share execution units, reducing kernel launch overhead. It also integrates FlashAttention/FlashInfer optimized kernels to enhance computational efficiency during scheduling further. To improve inference efficiency on edge devices, Yao *et al.* [22] proposed a direct compilation optimization scheme that aligns model compilation with target hardware instruction sets, reducing encoding latency and increasing decoding throughput.

Summary of system execution infrastructure

System implementations are key bridges for transforming algorithmic advantages into underlying execution efficiency. Hardware adaptation strategies demonstrate the customization potential for heterogeneous CPU, GPU, and NPU characteristics, but the lack of a unified standard across hardware platforms currently hinders smooth optimization migration for specific chips. Compilation layers and deployment frameworks (e.g., llama.cpp [160], MLC-LLM [162]) significantly reduce latency through instruction set alignment and kernel fusion, but still face conflicts between static graph optimization and dynamic multimodal token processing when handling large-scale multimodal inputs. In actual deployment, developers often need to strike a complex balance between native hardware support, framework generality, and the engineering costs of manual operator optimization.

Collaborative processing

Co-processing primarily includes cloud-edge and multi-edge device co-processing, both optimizing computational performance, reducing latency, and lowering energy consumption through effective resource sharing and task allocation based on specific needs.

Cloud-edge collaboration

Cloud-edge collaborative processing offloads computational tasks to the cloud, effectively alleviating the resource constraints of edge devices and enhancing edge inference performance. Although data transmission may introduce some latency, optimizing task allocation and resource utilization can significantly improve overall processing efficiency.

For example, Rjoub *et al.* [180] proposed a cloud-edge federated learning framework that reduces communication costs by 30% and improves model accuracy in resource-constrained environments. Li *et al.* [181] split tasks between edge devices for image feature extraction and cloud servers for text embedding and LLM inference, using dynamic flow control to balance edge and cloud resource use. TinyVision [182] deploys visual encoding tasks on edge to generate compact feature tokens, reducing both computation and data transmission.

In intelligent transportation systems, Hu *et al.* [183] used a cloud-edge architecture in which the edge processed real-time perception tasks and the cloud handled complex reasoning. With dataset fine-tuning and a DDPG offloading algorithm, they optimized both inference efficiency and accuracy. In mobile automation, the EcoAgent [184] framework uses cloud-edge collaboration to balance task planning and UI interactions, reducing communication overhead while maintaining performance. The CD-CCA framework [185] selects key multimodal data on edge for cloud processing, improving model generalization and adapting to dynamic data.

Multi-edge collaboration

As edge device capabilities improve, multi-edge device co-processing has become critical for advancing MLLM applications. This approach enables multiple edge devices to share resources and collaborate on task execution, ensuring efficient processing. Rjoub *et al.* [180] applied the PSO and ACO algorithms in autonomous systems to select devices and optimize communication paths, achieving a model accuracy of 92% and reducing communication costs by 30%.

Summary of collaborative processing

Collaborative processing addresses the resource bottleneck of a single device by enabling flexible task distribution through cloud-edge collaboration or multi-edge cooperation. The cloud-edge model uses cloud computing for complex tasks but relies heavily on network stability and low latency, with potential data privacy risks. Multi-device collaboration improves decentralization and robustness, but synchronization costs, unstable connections, and complex load balancing often limit performance in edge environments. Future advancements should focus on minimizing communication costs and maximizing inference throughput in dynamic network topologies.

APPLICATIONS

The integration of edge computing with MLLMs has led to a new class of intelligent applications that offer low latency, high privacy, and context-awareness. Unlike traditional AI, which relies on cloud comput-

ing, edge-native MLLMs run directly on resource-constrained devices, employing specialized acceleration strategies to support high-frequency decision-making feedback. These applications are classified into two categories based on the interaction environment: cyber intelligence, which focuses on digital operations, and embodied intelligence, which focuses on interactions with the physical world.

Cyber intelligence

Cyber intelligence refers to the direct analysis and interaction of MLLMs deployed on local devices with digital environments, such as screen pixels, documents, and system logs. The core of this technology lies in transforming raw multimodal data into semantic understanding, which is then used to infer user intent and assist in decision-making or automate software operations.

Interactive human assistance

Edge-based inference is essential for maintaining low-latency interactions in human-centric tasks. LLaVA-Phi [40] and Xmodel-VLM [186] demonstrate the feasibility of deploying lightweight VLMs on consumer-grade edge devices, providing instantaneous visual question-answering and multimodal dialogue. To further optimize task efficiency, T3-Agent [187] implements a unified scheduling mechanism for tool calls on edge, reducing the latency of multi-step problem-solving. Specialized tasks, such as the structured parsing of text-heavy documents, are addressed by KOSMOS-2.5-CHAT [188], which enables local, high-fidelity information retrieval without cloud dependency.

Autonomous digital agents

Beyond passive support, MLLMs act as autonomous agents to execute complex operations within digital ecosystems. Qwen2-VL [77] utilizes dynamic resolution and structural optimizations to navigate UIs and execute code interpreters on edge. Similarly, the AppAgent series [189, 190] empowers devices to autonomously explore and operate Android applications across varied scenarios. For high-frequency mobile tasks like real-time map navigation, Mobile-Agent [191] integrates OCR and grounding models to execute planning on-device. For desktop environments, MMAC-Copilot [192] employs a modular collaborative framework to automate complex GUI operations through edge-side planning and code generation. For long-form video analysis, AVA [193] leverages VLMs as autonomous agents, structuring video content via event knowledge graphs and enabling open-domain multimodal inference.

Embodied intelligence

Embodied intelligence extends MLLM intelligence into the physical realm by generating real-time control commands for hardware entities. These applications demand deterministic low latency to ensure the safety and stability of closed-loop control.

Autonomous driving

Edge-deployed MLLMs must process multi-source sensor data within millisecond windows for driving safety [194]. EM-VLM4AD [195] achieves real-time traffic scene understanding by reducing computa-

tional requirements to less than 10% of traditional models, making it suitable for vehicular edge platforms. The DriveGPT4 series [196, 197] provides end-to-end explainable driving by integrating vehicle states with multi-view vision for on-road decision-making. Furthermore, PlanAgent [198] enhances motion trajectory generation by fusing BEV (Bird's Eye View) information. Recent advances also explore collaborative intelligence. Rjoub *et al.* [180] proposed a federated learning framework for obstacle recognition, while Hu *et al.* [183] utilized edge-side LoRA fine-tuning to assist cloud-based LLMs in complex lane-detection tasks.

Robotic manipulation

Robotics applications require ultra-low latency for spatial perception and gait control. To optimize inference costs, DeeR-VLA [157] employs a dynamic early-exit mechanism, allowing robots to generate actions efficiently. For high-frequency operations, QUART-Online [199] delivers 50 Hz decision-making for quadrupedal navigation. Spatial awareness is further enhanced by RoboTron-Mani [200] and SC-MLLM [201], which provide 6D control command generation and self-correcting pose prediction on edge. In open-world scenarios, Magma [202] and OWMM-Agent [203] enable multi-degree-of-freedom manipulation and mobile grasping. Finally, VeBrain [204] leverages edge-cloud collaboration to transform visual tasks into text-based planning, facilitating low-sample generalization for complex robotic transport tasks.

FUTURE DIRECTIONS

Deploying MLLMs on edge offers unprecedented opportunities for real-time interactive applications [205, 206]. However, due to the limited computational power of edge devices, strict power consumption constraints, and the heterogeneity of storage hierarchies, efficient inference remains a significant challenge. Single-model compression or simple computation scheduling strategies are no longer sufficient to meet the growing performance demands [207]. Future research trends are moving towards the deep integration of algorithms, hardware, and systems. Based on this, this paper explores the following key research directions and challenges.

(i) Edge-native multimodal architectures. Currently, most multimodal models follow a cloud-based training and edge compression approach. This top-down methodology often overlooks the practical constraints of the underlying hardware. A key future research direction is to design edge-native MLLM architectures that optimize the architecture under hardware limitations to achieve deep cross-modal integration, rather than merely stacking features. This includes developing ultra-sparse MoE models [208] that can efficiently integrate image, text, and audio information, as well as linear attention mechanisms optimized for low-power environments, aiming to minimize the consumption of computational resources and energy on edge while ensuring inference accuracy.

(ii) Dynamic perception and adaptive inference framework. The complexity of edge environments requires inference systems to possess real-time perception and intelligent tuning capabilities. For example, MMEdge [209] actively implements dynamic balancing through pipelined sensing and encoding with its adaptive configuration optimizer. Future developments will include the incorporation of online reinforcement learning mechanisms to dynamically select model quantization rates, inference path depths, and decisions on speculative skipping, thereby enabling real-time optimal balancing between changing latency requirements

and device states.

(iii) Deep-layer hardware integration. Innovations at the hardware level directly determine the upper limits of inference acceleration, with the core trend focusing on building dedicated computing systems centered around the transformer mechanism and guided by storage optimization. First, to address the computational and memory challenges of the attention mechanism, the introduction of a Dedicated memory management unit (MMU) effectively manages *KV* caches [210], significantly alleviating memory bandwidth bottlenecks. Second, to tackle the energy consumption issues associated with data movement, processing-in-memory (PIM) technology embeds computation tasks directly within the storage [211, 212], enabling efficient processing close to the memory. Finally, through the deep integration of heterogeneous computing architectures and dedicated accelerators such as NPU-specific designs [213], hardware can achieve dynamic load balancing based on the features of multimodal tasks. This targeted optimization, driven by underlying hardware characteristics, ensures that complex large models can still achieve low power consumption and high real-time inference performance on resource-constrained edge devices.

(iv) Operating system-level deep integration. As MLLMs become fundamental services for edge devices, deeply integrating large model inference with the operating system will be critical. This involves efficiently scheduling inference tasks, managing resource isolation, and enabling multi-tenancy at the kernel level. Additionally, to address the high engineering costs of manual co-design, full-stack automation will become mainstream. Tools like EdgeTran [214] will enable joint search across model architecture, compiler strategies, and hardware micro-architecture, automatically generating optimal end-to-end acceleration solutions from the algorithmic layer to the hardware layer.

CONCLUSION

This survey reviews the progress of edge MLLMs, focusing on model-level and system-level optimization challenges. As MLLMs advance, edge deployment becomes essential for practical applications, particularly in visual, linguistic, and speech modalities. We cover model compression techniques, including visual token compression, visual token dropping, sparse attention, and cross-modal connectors, alongside strategies like distillation, quantization, and pruning to enhance inference efficiency. At the system level, we explore computation scheduling, hardware adaptation, and collaborative processing to address resource limitations and high latency. We also examine applications in cyber intelligence and embodied intelligence. Despite existing challenges, advancements in edge-native multimodal architectures, dynamic inference frameworks, deep-layer hardware integration, and operating system-level optimization offer promising directions for the future development of edge MLLMs. This survey aims to provide insights for researchers and support the wider application of MLLMs on edge.

Funding

This work was supported by the Fundamental and Interdisciplinary Disciplines Breakthrough Plan of the Ministry of Education of China (JYB2025XDXM116), the Mobile Information Networks National Science and Technology Major Project (2026ZD1306000), the National Natural Science Foundation of China (NSFC) (92467301), and the Key Research and Development Program of Zhejiang Province (2025C01012).

Author contributions

S.C. designed the research and wrote the initial manuscript. Y.S. and C.W. provided the source supports and supervision. All authors have contributed to the discussion of the content and have revised and edited the manuscript.

Conflict of interest

The authors declare no conflict of interest.

References

1 Vaswani A, Shazeer N, Parmar N, *et al*. Attention is all you need. In: *Proceedings of the Annual Conference on Neural Information Processing Systems*. Long Beach, CA, 2017, 5998–6008.

2 Raffel C, Shazeer N, Roberts A, *et al*. Exploring the limits of transfer learning with a unified text-to-text transformer. J Mach Learn Research 2020; **21**: 1–67.

3 Radford A, Narasimhan K, Salimans T, *et al*. Improving language understanding by generative pre-training. OpenAI Technical Report 2018. https://www.mikecaptain.com/resources/pdf/GPT-1.pdf.

4 Radford A, Wu J, Child R, *et al*. Language models are unsupervised multitask learners. OpenAI Blog 2019, 1: 8–9.

5 Brown T, Mann B, Ryder N, *et al*. Language models are few-shot learners. In: *Proceedings of the International Conference on Neural Information Processing Systems*. Vancouver BC, 2020, 1877–1901.

6 Devvlin J, Chang M-W, Lee K, *et al*. Bert: Pre-training of deep bidirectional transformers for language understanding. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Minneapolis, MN, 2019, 4171–4186.

7 Radford A, Kim J W, Hallacy C, *et al*. Learning transferable visual models from natural language supervision. In: *Proceedings of the International Conference on Machine Learning*. Virtual, 2021, 8748–8763.

8 Li J, Li D, Xiong C, *et al*. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In: *Proceedings of the International Conference on Machine Learning*. Baltimore, 2022.

9 Zhang C, Yang Z, He X, *et al*. Multimodal intelligence: Representation learning, information fusion, and applications. *IEEE J Sel Top Signal Process* 2020; **14**: 478–493.

10 Zhao F, Zhang C, Geng B. Deep multimodal data fusion. arXiv: 2312.11805.

11 Li Y, Jiang S, Hu B, *et al*. Uni-MoE: Scaling unified multimodal LLMs with mixture of experts. *IEEE Trans Pattern Anal Mach Intell* 2025; **47**: 3424–3439.

12 Ye Q, Xu H, Ye J, *et al*. mPLUG-Owl2: Revolutionizing multi-modal large language model with modality collaboration. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. Seattle, WA, 2024, 13040–13051.

13 Alayrac J-B, Donahue J, Luc P, *et al*. Flamingo: A visual language model for few-shot learning. In: *Proceedings of the Annual Conference on Neural Information Processing Systems*. New Orleans, LA, 2022, 23789–23803.

14 Driess D, Xia F, Sajjadi MSM, *et al*. PaLM-E: An embodied multimodal language model. In: *Proceedings of the International Conference on Machine Learning*. Honolulu, 2023.

15 Li J, Li D, Savarese S, *et al*. BLIP-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In: *Proceedings of the International Conference on Machine Learning*. Honolulu, 2023.

16 Yang Z, Li L, Lin K, *et al*. The dawn of lmms: Preliminary explorations with GPT-4V (ision). arXiv: 2309.17421.

17 Gemini Team, Anil R, Borgeaud S, *et al*. Gemini: A family of highly capable multimodal models. arXiv: 2312.11805.

18 Bourechak A, Zedadra O, Kouahla MN, *et al*. At the confluence of artificial intelligence and edge computing in IoT-based applications: A review and new perspectives. *Sensors* 2023; **23**: 1639.

19 Wang X, Garg S, Lin H, *et al*. A secure data aggregation strategy in edge computing and blockchain-empowered internet of things. *IEEE Internet Things J* 2020; **9**: 14237–14246.

20 Alwarafy A, Al-Thelaya KA, Abdallah M, *et al*. A survey on security and privacy issues in edge-computing-assisted

internet of things. *IEEE Internet Things J* 2020; **8**: 4004–4022.

21 Al-Ansi A, Al-Ansi AM, Muthanna A, *et al*. Survey on intelligence edge computing in 6G: Characteristics, challenges, potential use cases, and market drivers. *Future Internet* 2021; **13**: 118.

22 Yao Y, Yu T, Zhang A, *et al*. Efficient GPT-4V level multimodal large language model for deployment on edge devices. *Nat Commun* 2025; **16**: 5509.

23 Kaplan J, McCandlish S, Henighan T, *et al*. Scaling laws for neural language models. arXiv: 2001.08361.

24 Gholami A, Yao Z, Kim S, *et al*. AI and memory wall. *IEEE Micro* 2024; **44**: 33–39.

25 Zheng Y, Chen Y, Qian B, *et al*. A review on edge large language models: Design, execution, and applications. *ACM Comput Surv* 2025; **57**: 1–35.

26 Li J, Li J, Yang G, *et al*. Applications of large language models and multimodal large models in autonomous driving: A comprehensive review. *Drones* 2025; **9**: 238.

27 Bajpai K, Gupta V. EcoLLM: A joint optimization framework for ultra-low power, mixed-precision LLM inference on resource-constrained edge systems. 2025, 10.36227techrxiv.176114085.56648160/v1.

28 Xiong ZB, Luo XR, Wang BN. Research on lightweight deployment of artificial intelligence models and intelligent network bandwidth optimization based on edge computing. In: *Proceedings of the 2025 IEEE 8th International Conference on Automation, Electronics and Electrical Engineering (AUTEEE)*. Shenyang, 2025.

29 Chen Y, Yan Y, Ge S, *et al*. Confidant: Customizing transformer-based LLMs via collaborative training on mobile devices. In: *Proceedings of the 31st Annual International Conference on Mobile Computing and Networking*. Hong Kong, 2025, 483–497.

30 Jin Y, Li J, Liu Y, *et al*. Efficient multimodal large language models: A survey. arXiv: 2405.10739.

31 Zhou Z, Ning X, Hong K, *et al*. A survey on efficient inference for large language models. arXiv: 2404.14294.

32 Sharshar A, Khan LU, Ullah W, *et al*. Vision-language models for edge networks: A comprehensive survey. *IEEE Internet Things J* 2025; **12**: 32701–32724.

33 Li Y, Liu Z, Li Z, *et al*. Perception, reason, think, and plan: A survey on large multimodal reasoning models. arXiv: 2505.04921.

34 Han J, Gong K, Zhang Y, *et al*. Onellm: One framework to align all modalities with language. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. Seattle, WA, 2024.

35 Xing S, Qian C, Wang Y, *et al*. OpenEMMA: Open-source multimodal model for end-to-end autonomous driving. In: *Proceedings of the 2025 IEEE/CVF Winter Conference on Applications of Computer Vision Workshops (WACVW)*, Tucson, AZ, 2025, 911–919.

36 Luo R, Lin T-E, Zhang H, *et al*. OpenOmni: Advancing open-source omnimodal large language models with progressive multimodal alignment and real-time emotional speech synthesis. In: *Proceedings of the Annual Conference on Neural Information Processing Systems*. San Diego, CA, 2025.

37 Cantini R, Orsino A, Talia D. Xai-driven knowledge distillation of large language models for efficient deployment on low-resource devices. *J Big Data* 2024; **11**: 63.

38 Liu H, Li C, Wu Q, *et al*. Visual instruction tuning. In: *Proceedings of the Annual Conference on Neural Information Processing Systems*. New Orleans, LA, 2023.

39 Chu X, Qiao L, Lin X, *et al*. Mobilevlm: A fast, strong and open vision language assistant for mobile devices. arXiv: 2312.16886.

40 Zhu Y, Zhu M, Liu N, *et al*. LLaVA-Phi: Efficient multi-modal assistant with small language model. In: *Proceedings of the 1st International Workshop on Efficient Multimedia Computing under Limited*. Melbourne VIC, 2024, 18–22.

41 Zhu M, Zhu Y, Liu X, *et al*. Mipha: A comprehensive overhaul of multimodal assistant with small language models. arXiv: 2403.06199.

42 Javaheripi M, Bubeck S, Abdin M, *et al*. Phi-2: The surprising power of small language models. Microsoft Res Blog 2023, 1: 3–3.

43 Zhang P, Zeng G, Wang T, *et al*. Tinyllama: An open-source small language model. arXiv: 2401.02385.

44 Yuan Z, Li Z, Huang W, *et al*. TinyGPT-V: Efficient multimodal large language model via small backbones. In: *Proceedings of the 2nd Workshop on Advancing Neural Network Training: Computational Efficiency, Scalability, and Resource Optimization (WANT@ ICML 2024)*. Vienna, 2024.

45 Wei H, Kong L, Chen J, *et al*. Small language model meets with reinforced vision vocabulary. arXiv: 2401.12503.

46 Google. Gemma 3n. 2025. https://ai.google.dev/gemma/docs/gemma-3n.

47 Yao Y, Yu T, Zhang A, *et al*. Minicpm-v: A GPT-4V level mllm on your phone. arXiv: 2408.01800.

48 OpenBMB MiniCPM-o Team. Minicpm-o 2.6: A GPT-4O level mllm for vision, speech, and multimodal live streaming on your phone. 2025. https://openbmb.notion.site/MiniCPM-o-2-6-A-GPT-4o-Level-MLLM-for-Vision-Speech-and-Multimodal-Live-Streaming-on-Your-Phone-185ede1b7a558042b5d5e45e6b237da9.

49 Yu T, Wang Z, Wang C, *et al*. Minicpm-v 4.5: Cooking efficient mllms via architecture, data, and training recipe. arXiv: 2509.18154.

50 ZhipuAI. GLM-Edge. 2024. https://github.com/zai-org/GLM-Edge.

51 Ning Z, Zhao J, Jin Q, *et al*. Inf-MLLM: Efficient streaming inference of multimodal large language models on a single GPU. arXiv: 2409.09086.

52 Lin Z, Lin M, Lin L, *et al*. Boosting multimodal large language models with visual tokens withdrawal for rapid inference. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Philadelphia, 2025, 105–113.

53 Singh G, Wang X, Hu Y, *et al*. Efficiently serving large multimodal models using epd disaggregation. In: *Proceedings of the International Conference on Machine Learning*. Vancouver, 2025.

54 Wang H, Yu Z, Spadaro G, *et al*. Folder: Accelerating multi-modal large language models with enhanced performance. arXiv: 2501.02430.

55 Xie X, Zhang X, Tang X, *et al*. MACTFusion: Lightweight cross transformer for adaptive multimodal medical image fusion. *IEEE J Biomed Health Inform* 2024; **29**: 3317–3328.

56 Wang J, Chen H, Zhang X, *et al*. CaPaT: Cross-aware paired-affine transformation for multimodal data fusion network. *IEEE Geosci Remote Sens Lett* 2025; **22**: 1–5.

57 Yu J, Zhou S, Yang D, *et al*. Mquant: Unleashing the inference potential of multimodal large language models via static quantization. In: *Proceedings of the 33rd ACM International Conference on Multimedia*. Dublin, 2025, 1783–1792.

58 Gagrani M, Goel R, Jeon W, *et al*. On speculative decoding for multimodal large language models. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. Seattle, WA, 2024.

59 Shukor M, Cord M. Skipping computations in multimodal LLMs. arXiv: 2410.09454.

60 Wan Z, Wu Z, Liu C, *et al*. Look-m: Look-once optimization in KV cache for efficient multimodal long-context inference. arXiv: 2406.18139.

61 Huang W, Zhai Z, Shen Y, *et al*. Dynamic-LLaVA: Efficient multimodal large language models via dynamic vision-language context sparsification. In: *Proceedings of the Thirteenth International Conference on Learning Representations*. Singapore, 2025.

62 Pope R, Douglas S, Chowdhery A, *et al*. Efficiently scaling transformer inference. Proceed Mach Learn Sys 2023, 5: 606–624.

63 NVIDIA NIM LLMs Benchmarking. 2025. https://docs.nvidia.com/nim/benchmarking/llm/latest/metrics.html.

64 Desislavov R, Martínez-Plumed F, Hernández-Orallo J. Trends in AI inference energy consumption: Beyond the performance-vs-parameter laws of deep learning. *Sustain Computing-Inf Syst* 2023; **38**: 100857.

65 Kwon W, Li Z, Zhuang S, *et al*. Efficient memory management for large language model serving with pagedattention. In: *Proceedings of the 29th Symposium on Operating Systems Principles*. Koblenz, 2023, 611–626.

66 Sheng Y, Zheng L, Yuan B, *et al*. Flexgen: High-throughput generative inference of large language models with a single GPU. In: *Proceedings of the International Conference on Machine Learning*. Honolulu 2023, 202: 31094–31116.

67 Fu C, Chen P, Shen Y, *et al*. MME: A comprehensive evaluation benchmark for multimodal large language models.

arXiv: 2306.13394.

68 Liu Y, Duan H, Zhang Y, *et al*. MMBench: Is your multi-modal model an all-around player? In: *Computer Vision—ECCV 2024. ECCV 2024. Lecture Notes in Computer Science*. Cham: Springer, 2024.

69 Yue X, Ni Y, Zhang K, *et al*. Mmmu: A massive multi-discipline multimodal understanding and reasoning benchmark for expert agi. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. Seattle, WA, 2024.

70 Lu P, Bansal H, Xia T, *et al*. Mathvista: Evaluating mathematical reasoning of foundation models in visual contexts. In: *Proceedings of the Twelfth International Conference on Learning Representations*. Vienna, 2024.

71 Fu C, Dai Y, Luo Y, *et al*. Video-MME: The first-ever comprehensive evaluation benchmark of multi-modal llms in video analysis. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. Nashville, TN, 2025.

72 Yang N, Wen J, Zhang M, *et al*. Generalizable pareto-optimal offloading with reinforcement learning in mobile edge computing. *IEEE Trans Serv Comput* 2025; **18**: 3824–3836.

73 Shao K, Tao K, Zhang K, *et al*. When tokens talk too much: A survey of multimodal long-context token compression across images, videos, and audios. arXiv: 2507.20198.

74 Ma Y, Abdelraouf A, Gupta R, *et al*. Video token sparsification for efficient multimodal LLMs in driving visual question answering. In: *Proceedings of the 2025 IEEE Intelligent Vehicles Symposium (IV)*. Cluj-Napoca, 2025.

75 Gao Z, Wang Y, Chen J, *et al*. MMTSA: Multi-modal temporal segment attention network for efficient human activity recognition. In: *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*. 2023, 1–26.

76 Gao Z, Chen Z, Cui E, *et al*. Mini-InternVL: A flexible-transfer pocket multi-modal model with 5% parameters and 90% performance. *Vis Intell* 2024; **2**: 32.

77 Wang P, Bai S, Tan S, *et al*. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. arXiv: 2409.12191.

78 Su J, Ahmed M, Lu Y, *et al*. RoFormer: Enhanced transformer with Rotary Position Embedding. *Neurocomputing* 2024; **568**: 127063.

79 Kimi Team, Du A, Yin B, *et al*. Kimi-VL technical report. arXiv: 2504.07491.

80 Guo Z, Xu R, Yao Y, *et al*. LLaVA-UHD: An LMM perceiving any aspect ratio and high-resolution images. In: *Computer Vision—ECCV 2024. ECCV 2024. Lecture Notes in Computer Science*. Cham: Springer, 2024.

81 Dehghani M, Mustafa B, Djolonga J, *et al*. Patch n'Pack: Navit, a vision transformer for any aspect ratio and resolution. In: *Proceedings of the Annual Conference on Neural Information Processing Systems*. New Orleans, LA, 2023.

82 Zhai X, Mustafa B, Kolesnikov A, *et al*. Sigmoid loss for language image pre-training. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. Paris, 2023.

83 Li B, Zhang Y, Guo D, *et al*. Llava-onevision: Easy visual task transfer. arXiv: 2408.03326.

84 Xiong B, Chen B, Wang C, *et al*. BlueLM-2.5-3B Technical Report. arXiv: 2507.05934.

85 Vasu PKA, Faghri F, Li C-L, *et al*. Fastvlm: Efficient vision encoding for vision language models. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Nashville, TN, 2025.

86 Bai S, Chen K, Liu X, *et al*. Qwen2.5-VL technical report. arXiv: 2502.13923.

87 Huang M, Huang R, Shi H, *et al*. Efficient multi-modal large language models via visual token grouping. arXiv: 2411.17773.

88 Marafioti A, Zohar O, Farré M, *et al*. Smolvlm: Redefining small and efficient multimodal models. arXiv: 2504.05299.

89 Yang G, Yan X, Kou H, *et al*. TWDP: A vision transformer accelerator with token-weight dual-pruning strategy for edge device deployment. In: *Proceedings of the 30th Asia and South Pacific Design Automation Conference*. Tokyo, 2025, 177–182.

90 Zhao S, Wang Z, Juefei-Xu F, *et al*. Accelerating multimodal large language models by searching optimal vision token reduction. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Nashville, TN, 2025.

91 Tang Y, Han K, Wang Y, *et al*. Patch slimming for efficient vision transformers. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. New Orleans, LA, 2022.

92 Chen L, Zhao H, Liu T, *et al*. An image is worth 1/2 tokens after layer 2: Plug-and-play inference acceleration for large vision-language models. In: *Computer Vision—ECCV 2024. ECCV 2024. Lecture Notes in Computer Science*. Cham: Springer, 2024.

93 Liu H, Li C, Li Y, *et al*. Improved baselines with visual instruction tuning. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. Seattle, WA, 2024.

94 Bai J, Bai S, Yang S, *et al*. Qwen-VL: A versatile vision-language model for understanding, localization, text reading, and beyond. arXiv: 2308.12966.

95 Han Y, Liu X, Ding P, *et al*. Rethinking token reduction in MLLMs: Towards a unified paradigm for training-free acceleration. arXiv: 2411.17686.

96 Sun Y, Xin Y, Li H, *et al*. LVPruning: An effective yet simple language-guided vision token pruning approach for multi-modal large language models. In: *Findings of the Association for Computational Linguistics: NAACL 2025*. Albuquerque: Association for Computational Linguistics, 2025, 4299–4308.

97 Zhao Z, Li Y, Li Y. Learning free token reduction for multi-modal large language models. arXiv: 2501.17391.

98 Lin B, Ye Y, Zhu B, *et al*. Video-LLaVA: Learning united visual representation by alignment before projection. In: *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing* 2024.

99 Liu T, Shi L, Hong R, *et al*. Multi-stage vision token dropping: Towards efficient multimodal large language model. arXiv: 2411.10803.

100 Liu H, Li C, Li Y, *et al*. LLaVA-NeXT: Improved reasoning, OCR, and world knowledge. 2024. https://llava-vl.github.io/blog/2024-01-30-llava-next/.

101 Singh A, Natarajan V, Shah M, *et al*. Towards VQA models that can read. In: *Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. Long Beach, CA, 2019.

102 Gholami M, Akbari M, Cannons K, *et al*. CASP: Compression of large multimodal models based on attention sparsity. In: *Proceedings of the 2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Nashville, TN, 2025.

103 Chen F, He Y, Lin L, *et al*. Zipr1: Reinforcing token sparsity in MLLMs. arXiv: 2504.18579.

104 Mitra C, Huang B, Chai T, *et al*. Enhancing few-shot vision-language classification with large multimodal model features. In: *Proceedings of the 2025 IEEE/CVF International Conference on Computer Vision (ICCV)*. Honolulu, HI, 2025.

105 Chu Y F, Xu J, Yang Q, *et al*. Qwen2-audio technical report. arXiv: 2407.10759.

106 Yang T, Ma F, Li X, *et al*. DTATrans: Leveraging dynamic token-based quantization with accuracy compensation mechanism for efficient transformer architecture. *IEEE Trans Comput-Aided Des Integr Circuits Syst* 2022; **42**: 509–520.

107 Jia D, Guo J, Han K, *et al*. GeminiFusion: Efficient pixel-wise multimodal fusion for vision transformer. In: *Proceedings of the International Conference on Machine Learning*. Vienna, 2024.

108 Li B, Li Y, Li Z, *et al*. Megrez-omni technical report. arXiv: 2502.15803.

109 Chu X, Qiao L, Zhang X, *et al*. Mobilevlm v2: Faster and stronger baseline for vision language model. arXiv: 2402.03766.

110 Zhang S, Fang Q, Yang Z, *et al*. Llava-mini: Efficient image and video large multimodal models with one vision token. In: *Proceedings of the Thirteenth International Conference on Learning Representations, ICLR 2025*. Singapore, 2025.

111 Zhu D, Chen J, Shen X, *et al*. MiniGPT-4: Enhancing vision-language understanding with advanced large language models. In: *Proceedings of the Twelfth International Conference on Learning Representations*. Vienna, 2024.

112 Li W, Zhou H, Yu J, *et al*. Coupled mamba: Enhanced multimodal fusion with coupled state space model. In:

Proceedings of the Annual Conference on Neural Information Processing Systems. Vancouver, 2024.

113 Sun X, Yang Z, Xie R, *et al*. LightVLP: A lightweight vision-language pre-training via gated interactive masked autoencoders. In: *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*. Torino, 2024, 10499–10510.

114 Hu Y, Fan Z, Wang X, *et al*. TinyAlign: Boosting lightweight vision-language models by mitigating modal alignment bottlenecks. arXiv: 2505.12884.

115 Goyal Y, Khot T, Summers-Stay D, *et al*. Making the V in VQA matter: Elevating the role of image understanding in visual question answering. In: *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Honolulu, HI, 2017.

116 Cai Z, Cao M, Chen H, *et al*. InternLM2 technical report. arXiv: 2403.17297.

117 Abdin M I, Jacobs S A, Awan A A, *et al*. Phi-3 technical report: A highly capable language model locally on your phone. arXiv: 2404.14219.

118 Bellagente M, Tow J, Mahan D, *et al*. Stable LM 2.1.6B technical report. arXiv: 2402.17834.

119 Bai J, Bai S, Chu Y, *et al*. Qwen technical report. arXiv: 2309.16609.

120 Biderman S, Schoelkopf H, Anthony QG, *et al*. Pythia: A suite for analyzing large language models across training and scaling. In: *Proceedings of the International Conference on Machine Learning*. Honolulu 2023.

121 Liu Y, Li Z, Huang M, *et al*. OCRBench: On the hidden mystery of OCR in large multimodal models. *Sci China Inf Sci* 2024; **67**: 220102.

122 Mathew M, Karatzas D, Jawahar CV. DocVQA: A dataset for VQA on document images. In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. Online, 2021.

123 Yu L, Poirson P, Yang S, *et al*. Modeling context in referring expressions. In: *Computer Vision—ECCV 2016. ECCV 2016. Lecture Notes in Computer Science*. Cham: Springer, 2016.

124 Zhang B, Sennrich R. Root mean square layer normalization. In: *Proceedings of the Annual Conference on Neural Information Processing Systems*. Vancouver, 2019.

125 Jiang A Q, Sablayrolles A, Roux A, *et al*. Mixtral of experts. arXiv: 2401.04088.

126 Lin B, Tang Z, Ye Y, *et al*. MoE-LLaVA: Mixture of experts for large vision-language models. IEEE T Multimedia, 2026: 1–14.

127 Gu A, Dao T. MAMBA: Linear-time sequence modeling with selective state spaces. In: *Proceedings of the First conference on language modeling*. Philadelphia, 2024.

128 Zhao H, Zhang M, Zhao W, *et al*. Cobra: Extending mamba to multi-modal large language model for efficient inference. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Philadelphia, 2025, 118–126.

129 Huang W, Pan J, Tang J, *et al*. Ml-MAMBA: Efficient multi-modal large language model utilizing MAMBA-2. arXiv: 2407.19832.

130 Dao T, Gu A. Transformers are SSMs: Generalized models and efficient algorithms through structured state space duality. In: *Proceedings of the 41st International Conference on Machine Learning (ICML)*. Vienna, 2024.

131 Thomas A, Massaroli S, Poli M. Liquid Edge Team. Convolutional multi-hybrids for edge devices. 2025. https://www.liquid.ai/research/convolutional-multi-hybrids-for-edge-devices.

132 Hou H, Zeng P, Ma F, *et al*. VisualRWKV: Exploring recurrent neural networks for visual language models. In: *Proceedings of the 31st International Conference on Computational Linguistics*. Abu Dhabi, 2025, 10423–10434.

133 Cai Y, Zhang J, He H, *et al*. LLaVA-KD: A framework of distilling multimodal large language models. In: *Proceedings of the 2025 IEEE/CVF International Conference on Computer Vision (ICCV)*. Honolulu, 2025.

134 Yang A, Li A, Yang B, *et al*. Qwen3 technical report. arXiv: 2505.09388.

135 Xu S, Li X, Yuan H, *et al*. LLaVADI: What matters for multimodal large language models distillation. arXiv: 2407.19409.

136 Feng Q, Li W, Lin T, *et al*. Align-KD: Distilling cross-modal alignment knowledge for mobile vision-language large

model enhancement. In: *Proceedings of the 2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Nashville, TN, 2025.

137 Cai K, Duan Z, Liu G, *et al*. Self-adapting large visual-language models to edge devices across visual modalities. In: *Computer Vision—ECCV 2024. Lecture Notes in Computer Science*. Cham: Springer, 2024.

138 Liao B, Tao H, Zhang Q, *et al*. Multimodal mamba: Decoder-only multimodal state space model via quadratic to linear distillation. arXiv: 2502.13145.

139 Gerganov G. GGML. 2024. https://github.com/ggerganov/ggml.

140 Koska B, Horváth M. Towards multi-modal mastery: A 4.5B parameter truly multi-modal small language model. In: *Proceedings of the 2024 2nd International Conference on Foundation and Large Language Models (FLLM)*. Dubai, 2024.

141 Chen Z, Wang W, Cao Y, *et al*. Expanding performance boundaries of open-source multimodal models with model, data, and test-time scaling. arXiv: 2412.05271.

142 Lin H, Bai H, Liu Z, *et al*. Mope-clip: Structured pruning for efficient vision-language models with module-wise pruning error metric. In: *Proceedings of the 2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. Seattle, WA, 2024.

143 Wu Z, Chen J, Wang Y. Unified knowledge maintenance pruning and progressive recovery with weight recalling for large vision-language models. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Philadelphia, 2025.

144 Liang Y, Wang Z, Xu X, *et al*. Efficientllava: Generalizable auto-pruning for large vision-language models. In: *Proceedings of the 2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Nashville, TN, 2025.

145 Zhang Z, Pan X, Wei H, *et al*. LOP: Learning optimal pruning for efficient on-demand MLLMs scaling. arXiv: 2506.12826.

146 Huang Y, Thede L, Mancini M, *et al*. Investigating structural pruning and recovery techniques for compressing multimodal large language models: An empirical study. Pattern Recognition. In: *DAGM GCPR 2025. Lecture Notes in Computer Science*. Cham: Springer, 2026.

147 Xiao G, Tian Y, Chen B, *et al*. Efficient streaming language models with attention sinks. In: *Proceedings of the Twelfth International Conference on Learning Representations*. Vienna, 2024.

148 Han I, Zhang Z, Wang Z, *et al*. CalibQuant: 1-Bit KV cache quantization for multimodal LLMs. In: *Proceedings of the ICML 2025 Workshop on Long-Context Foundation Models*. Vancouver, 2025.

149 Tillet P, Kung H-T, Cox D. Triton: An intermediate language and compiler for tiled neural network computations. In: *Proceedings of the 3rd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages*. Phoenix, AZ, 2019, 10–19.

150 Leviathan Y, Kalman M, Matias Y. Fast inference from transformers via speculative decoding. In: *Proceedings of the International Conference on Machine Learning*. Honolulu, 2023.

151 Miao X, Oliaro G, Zhang Z, *et al*. Specinfer: Accelerating large language model serving with tree-based speculative inference and verification. In: *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*. La Jolla, CA, 2024, 932–949.

152 Lin L, Lin Z, Zeng Z, *et al*. Speculative decoding reimagined for multimodal large language models. arXiv: 2505.14260.

153 Lu X, Chen Y, Chen C, *et al*. BlueLM-V-3B: Algorithm and system co-design for multimodal large language models on mobile devices. In: *Proceedings of the 2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Nashville, TN, 2025.

154 Dong X, Liu T, Zeng Y, *et al*. HydraInfer: Hybrid disaggregated scheduling for multimodal large language model serving. arXiv: 2505.12658.

155 Elhoushi M, Shrivastava A, Liskovich D, *et al*. Layerskip: Enabling early exit inference and self-speculative decoding.

In: *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Bangkok, 2024, 923–938.

156 Devvrit F, Kudugunta S, Kusupati A, *et al*. Matformer: Nested transformer for elastic inference. In: *Proceedings of the Workshop on Advancing Neural Network Training: Computational Efficiency, Scalability, and Resource Optimization (WANT@ NeurIPS 2023)*, New Orleans, 2023.

157 Yue Y, Wang Y, Kang B, *et al*. Deer-vla: Dynamic inference of multimodal large language models for efficient robot execution. In: *Proceedings of the 38th International Conference on Neural Information Processing Systems*. Vancouver, 2024.

158 Kwon W, Li Z, Zhuang S, *et al*. vLLM. 2023. https://github.com/vllm-project/vllm.

159 Zheng L, Yin L, Xie Z, *et al*. Sglang: Efficient execution of structured language model programs. In: *Proceedings of the Annual Conference on Neural Information Processing Systems*. Vancouver, 2024.

160 Gerganov G. llama.cpp. 2023. https://github.com/ggerganov/llama.cpp.

161 NVIDIA. TensorRT-LLM. 2025. https://nvda.org.cn/TensorRT-LLM/.

162 MLC Team. MLC LLM. 2025. https://github.com/mlc-ai/mlc-llm.

163 Lv C, Niu C, Gu R, *et al*. Walle: An end-to-end, general-purpose, and large-scale production system for device-cloud collaborative machine learning. In: *Proceedings of the 16th USENIX Symposium on Operating Systems Design and Implementation (OSDI 22)*. Carlsbad, CA, 2022.

164 Shao Z, Yu Z, Yu J, *et al*. Imp: Highly capable large multimodal models for mobile devices. *IEEE Trans Multimedia* 2025; **27**: 2961–2974.

165 Abid A, Abdalla A, Abid A, *et al*. Gradio: Hassle-free sharing and testing of ML models in the wild. arXiv: 1906.02569.

166 Huang M, Shen A, Li K, *et al*. EdgeLLM: A highly efficient CPU-FPGA heterogeneous edge accelerator for large language models. *IEEE Trans Circuits Syst I* 2025; **72**: 3352–3365.

167 Kim H, Ye G, Wang N, *et al*. Exploiting intel advanced matrix extensions (AMX) for Large Language Model Inference. *IEEE Comput Arch Lett* 2024; **23**: 117–120.

168 Zhu Y, Lu H. Edge-side NPU inference optimization: Adaptation research of multimodal large models on qualcomm platforms. Intell Data Anal 2025; 30: 544–568.

169 Bai K, Ye L, Huang R, *et al*. EdgeMM: Multi-core CPU with heterogeneous AI-extension and activation-aware weight pruning for multimodal LLMs at edge. arXiv: 2505.10782.

170 Khronos Group. OpenCL—The Open Standard for Parallel Programming of Heterogeneous Systems. 2025. https://www.khronos.org/opencl/.

171 Fu Z, Ren J, Liu Y, *et al*. Hyperion: A generic and distributed mobile offloading framework on OpenCL. In: *Proceedings of the 20th ACM Conference on Embedded Networked Sensor Systems*. Boston Massachusetts, 2022, 607–621.

172 Jia F, Zhang D, Cao T, *et al*. CoDL: Efficient CPU-GPU co-execution for deep learning inference on mobile devices. In: *Proceedings of the 20th Annual International Conference on Mobile Systems, Applications and Services (MobiSys'22)*. Portland, 2022, 209–221.

173 Apple Inc. Apple unleashes M1. 2020. https://www.apple.com/newsroom/2020/11/apple-unleashes-m1/.

174 Shen Y, Wang ZC, Wang TY, *et al*. Hetero2Pipe: Pipelining multi-DNN inference on heterogeneous mobile processors under co-execution slowdown. In: *Proceedings of the 2025 IEEE 45th International Conference on Distributed Computing Systems (ICDCS)*. Glasgow, 2025.

175 Qualcomm Team. Qualcomm Snapdragon 8s Gen 3. 2024. https://www.qualcomm.com/news/releases/2024/03/qualcomm-brings-the-best-of-on-device-ai-to-more-smartphones-wit.

176 Lin J, Tang J M, Tang H T, *et al*. AWQ: Activation-aware weight quantization for on-device LLM compression and acceleration. In: *Proceedings of the Seventh Annual Conference on Machine Learning and Systems*. Santa Clara, 2024, 87–100.

177 Xiao G, Lin J, Seznec M, *et al*. Smoothquant: Accurate and efficient post-training quantization for large language models. In: *Proceedings of the International Conference on Machine Learning*. Honolulu, 2023, 38087–38099.

178 Ma X Y, Fang G F, Wang X C. LLM-Pruner: On the structural pruning of large language models. In: *Proceedings of the Annual Conference on Neural Information Processing Systems*. New Orleans, LA, 2023.

179 Gokhale S, Das D, Patwari R, *et al*. KV Pareto: Systems-level optimization of KV cache and model compression for long context inference. In: *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 5: Industry Track)*. Rabat, 2026.

180 Rjoub G, Elmekki H, Islam S, *et al*. A hybrid swarm intelligence approach for optimizing multimodal large language models deployment in edge-cloud-based federated learning environments. *Comput Commun* 2025; **237**: 108152.

181 Li Y, Gumaste D, Turkcan M K, *et al*. Distributed VLMs: Efficient vision-language processing through cloud-edge collaboration. In: *Proceedings of the 2025 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops)*. Washington DC, 2025.

182 Lou S, Ge S, Yu J, *et al*. TinyVision: Distributed vision-language model with efficiency and privacy for edge deployment. In: *Advanced Intelligent Computing Technology and Applications. ICIC 2025. Lecture Notes in Computer Science*. Singapore: Springer, 2025.

183 Hu Y, Ye D, Kang J, *et al*. A cloud-edge collaborative architecture for multimodal LLM-based advanced driver assistance systems in IoT networks. *IEEE Internet Things J* 2025; **12**: 13208–13221.

184 Yi B, Hu X, Chen Y, *et al*. EcoAgent: An efficient edge-cloud collaborative multi-agent framework for mobile automation. arXiv: 2505.05440.

185 Wang G, Liu J, Li C, *et al*. Cloud-device collaborative learning for multimodal large language models. In: *Proceedings of the 2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. Seattle, WA, 2024.

186 Xu W, Liu Y, He L, *et al*. Xmodel-vlm: A simple baseline for multimodal vision language model. arXiv: 2405.09215.

187 Gao Z, Zhang B, Li P, *et al*. Multi-modal agent tuning: Building a VLM-driven agent for efficient tool usage. In: *Proceedings of the Thirteenth International Conference on Learning Representations*. Singapore, 2025.

188 Lv T, Huang Y, Chen J, *et al*. Kosmos-2.5: A multimodal literate model. arXiv: 2309.11419.

189 Zhang C, Yang Z, Liu J, *et al*. AppAgent: Multimodal agents as smartphone users. In: *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*. Yokohama, 2025, 1–20.

190 Li Y, Zhang C, Yang W, *et al*. Appagent v2: Advanced agent for flexible mobile interactions. arXiv: 2408.11824.

191 Wang J, Xu H, Ye J, *et al*. Mobile-Agent: Autonomous multi-modal mobile device agent with visual perception. In: *Proceedings of the ICLR 2024 Workshop on Large Language Model (LLM) Agents*. Vienna, 2024.

192 Song Z, Li Y, Fang M, *et al*. Mmac-copilot: Multi-modal agent collaboration operating system copilot. arXiv: 2404.18074.

193 Yan Y, Jiang S, Cao T, *et al*. AVA: Towards agentic video analytics with vision language models. arXiv: 2505.00254.

194 Li X, Ma Y, Chen Y, *et al*. Priority optimization for autonomous driving systems to meet end-to-end latency constraints. In: *Proceedings of the 2024 IEEE Real-Time Systems Symposium (RTSS)*. York, 2024.

195 Gopalkrishnan A, Greer R, Trivedi M. Multi-frame, lightweight & efficient vision-language models for question answering in autonomous driving. arXiv: 2403.19838.

196 Xu Z, Zhang Y, Xie E, *et al*. DriveGPT4: Interpretable end-to-end autonomous driving via large language model. *IEEE Robot Autom Lett* 2024; **9**: 8186–8193.

197 Xu Z, Bai Y, Zhang Y, *et al*. DriveGPT4-V2: Harnessing large language model capabilities for enhanced closed-loop autonomous driving. In: *Proceedings of the 2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Nashville, TN, 2025.

198 Zheng Y, Xing Z, Zhang Q, *et al*. Planagent: A multi-modal large language agent for closed-loop vehicle motion planning. arXiv: 2406.01587.

199 Tong X, Ding P, Fan Y, *et al*. Quart-Online: Latency-free multimodal large language model for quadruped robot

learning. In: *Proceedings of the 2025 IEEE International Conference on Robotics and Automation (ICRA)*. Atlanta, GA, 2025.

200 Yan F, Liu F, Huang Y, *et al*. RoboTron-Mani: All-in-one multimodal large model for robotic manipulation. In: *Proceedings of the 2025 IEEE/CVF International Conference on Computer Vision (ICCV)*. Honolulu, HI, 2025.

201 Liu J, Li C, Wang G, *et al*. Self-corrected multimodal large language model for end-to-end robot manipulation. arXiv: 2405.17418.

202 Yang J, Tan R, Wu Q, *et al*. Magma: A foundation model for multimodal AI agents. In: *Proceedings of the 2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Nashville, TN, 2025.

203 Chen J, Liang H, Du L, *et al*. OWMM-Agent: Open world mobile manipulation with multi-modal agentic data synthesis. In: *Proceedings of the RSS 2025 Workshop: Mobile Manipulation: Emerging Opportunities & Contemporary Challenges*. Los Angeles, California, 2025.

204 Luo G, Yang G, Gong Z, *et al*. Visual embodied brain: Let multimodal large language models see, think, and control in spaces. arXiv: 2506.00123.

205 Lin Z, Qu G, Chen Q, *et al*. Pushing large language models to the 6G edge: Vision, challenges, and opportunities. *IEEE Commun Mag* 2025; **63**: 52–59.

206 Xu M, Niyato D, Kang J, *et al*. When large language model agents meet 6G networks: Perception, grounding, and alignment. *IEEE Wireless Commun* 2024; **31**: 63–71.

207 Msuya H, Maiseli BJ. Deep learning model compression techniques: Advances, opportunities, and perspective. *Tanzania J Eng Technol* 2023; **42**: 65–83.

208 Li D, Liu Y, Wu H, *et al*. Aria: An open multimodal native mixture-of-experts model. arXiv: 2410.05993.

209 Huang R, Yu M, Tsoi M, *et al*. MMEdge: Accelerating on-device multimodal inference via pipelined sensing and encoding. arXiv: 2510.25327.

210 Moradifirouzabadi A, Kang M. End-to-end acceleration of generative models with runtime regularized KV cache management. *IEEE J Emerg Sel Top Circuits Syst* 2025; **15**: 217–230.

211 Lee J, Ha S. Empowering edge devices with processing-in-memory for on-device language inference. *IEEE Embedded Syst Lett* 2025; **17**: 244–247.

212 Oliveira GF, Gomez-Luna J, Ghose S, *et al*. Accelerating neural network inference with processing-in-DRAM: From the edge to the cloud. *IEEE Micro* 2022; **42**: 25–38.

213 Xu D, Zhang H, Yang L, *et al*. Fast on-device LLM inference with NPUs. In: *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 1*. Rotterdam, 2025, 445–462.

214 Tuli S, Jha N K. EdgeTran: Co-designing transformers for efficient inference on mobile edge platforms. arXiv: 2303.13745.