Efficient Multimodal Large Language Models: A Survey

Yizhang Jin[1,2,*], Jian Li[1,*], Yexin Liu[3], Tianjun Gu[4], Kai Wu[1], Zhengkai Jiang[1], Muyang He[3], Bo Zhao[3], Xin Tan[4], Zhenye Gan[1], Yabiao Wang[1], Chengjie Wang[1], Lizhuang Ma[2]

[1]Youtu Lab, Tencent, [2]SJTU, [3]BAAI, [4]ECNU

Abstract

In the past year, Multimodal Large Language Models (MLLMs) have demonstrated remarkable performance in tasks such as visual question answering, visual understanding and reasoning. However, the extensive model size and high training and inference costs have hindered the widespread application of MLLMs in academia and industry. Thus, studying efficient and lightweight MLLMs has enormous potential, especially in edge computing scenarios. In this survey, we provide a comprehensive and systematic review of the current state of efficient MLLMs. Specifically, we summarize the timeline of representative efficient MLLMs, research state of efficient structures and strategies, and the applications. Finally, we discuss the limitations of current efficient MLLM research and promising future directions. Please refer to our GitHub repository for more details: https://github.com/lijiannuist/Efficient-Multimodal-LLMs-Survey.

1 Introduction

Large-scale pretraining, a leading approach in Artificial Intelligence(AI), has seen general-purpose models like large language and multimodal models outperform specialized deep learning models across many tasks. The remarkable abilities of Large Language Models (LLM) have inspired efforts to merge them with other modality-based models to enhance multimodal competencies. This concept is further supported by the remarkable success of proprietary models like OpenAI's GPT-4V [1] and Google's Gemini[2]. As a result, Multimodal Large Language Models (MLLMs) have emerged, including the mPLUG-Owl series[3, 4], InternVL [5], EMU [6], LLaVA [7], InstructBLIP [8], MiniGPT-v2 [9], and MiniGPT-4[10]. These models circumvent the computational cost of training from scratch by effectively leveraging the pre-training knowledge of each modality. MLLMs inherit the cognitive capabilities of LLMs, showcasing numerous remarkable features such as robust language generation and transfer learning abilities. Moreover, by establishing strong representational connections and alignments with other modality-based models, MLLMs can process inputs from multiple modalities, significantly broadening their application scope.

The success of MLLMs is largely attributed to the scaling law: the performance of an AI model improves as more resources, such as data, computational power, or model size, are invested into it. However, scalability comes at the cost of high resource demands, which hinders the development and deployment of large models. For example, the training of MiniGPT-v2 necessitates a total of over 800 GPU hours, as calculated based on NVIDIA A100 GPUs [9]. This imposes a substantial expense that is difficult for researchers outside of major enterprises to bear. Aside from training,

[1]* Equal contribution.

[2] Yizhang Jin is an intern in Tencent, and Jian Li is the project leader.

Preprint. Under review.

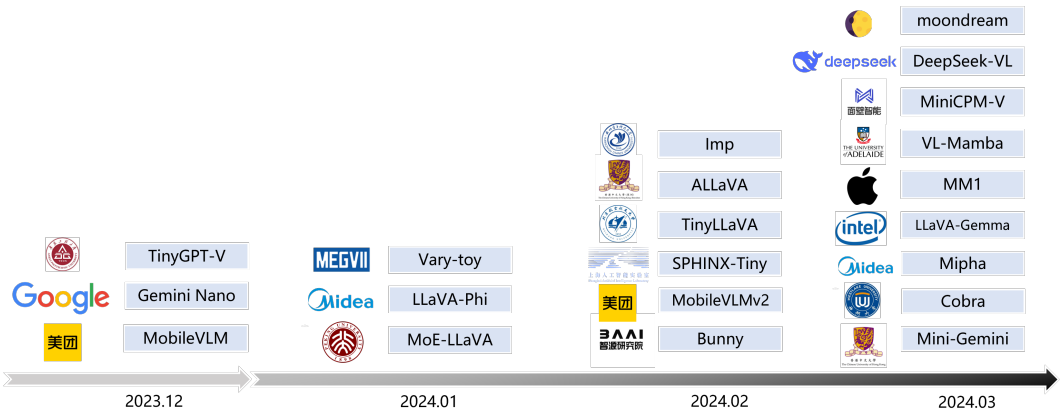


Figure 1: The timeline of efficient MLLMs.

inference constitutes the major portion of resource consumption in mllm. Consider a typical scenario where the model input consists of an image with dimensions of 336×336 pixels and a text prompt with a length of 40 tokens, performing inference with LLaVA-1.5 and a Vicuna-13B LLM backbone requires 18.2T FLOPS and 41.6G of memory usage. The resource-intensive nature of large-scale models has also sparked concerns about democratization and privacy protection, considering that the current mainstream MLLMs, represented by GPT-4V and Gemini, are controlled by a few dominant corporations and operate in the cloud. As demonstrated in the aforementioned experiments, even for open-source MLLMs, high requirements for computation resources make it challenging to run them on edge devices. This further exacerbates the challenges associated with ensuring equitable access and preserving user privacy.

In light of these challenges, there has been growing attention on the study of efficient MLLMs. The primary objective of these endeavors is to decrease the resource consumption of MLLMs and broaden their applicability while minimizing performance degradation. Research on efficient MLLMs began with replacing large language models with lightweight counterparts and performing typical visual instruction tuning. Subsequent studies further enhanced capabilities and expanded use cases in the following ways: (1) lighter architectures were introduced with an emphasis on efficiency, aiming to reduce the number of parameters or computational complexity[25, 13, 18]; (2) more specialized components were developed, focusing on efficiency optimizations tailored to advanced architectures or imbuing specific properties, such as locality[19, 17, 12]; and (3) support for resource-sensitive tasks was provided, with some works employing visual token compression to boost efficiency, enabling the transfer of MLLM capabilities to resource-intensive tasks such as high-resolution image and video understanding[35, 39, 14, 40].

In this survey, we aim to present an exhaustive organization of the recent advancements in the rapidly evolving field of efficient MLLMs, as depicted in Figure.2. We organize the literature in a taxonomy consisting of six primary categories, encompassing various aspects of efficient MLLMs, including **architecture**, **efficient vision**, **efficient LLMs**, **training**, **data and benchmarks**, and **applications**.

- **Architecture** focuses on the MLLM framework developed by efficient techniques to reduce the computational cost. The architecture is composed of multiple modality-based fundamental models, exhibits characteristics distinct from single-modal models, thus promoting the development of novel technologies.

- **Efficient Vision** explores optimizing efficient visual fracture extraction strategies, emphasizing methods that boost efficiency while maintaining accuracy. It addresses integrating high-quality visual data for effective cross-modal understanding.

- **Efficient LLMs** explores these strategies of improving the computational efficiency and scalability of language models. It examines the trade-offs between model complexity and performance while suggesting promising avenues for balancing these competing factors.

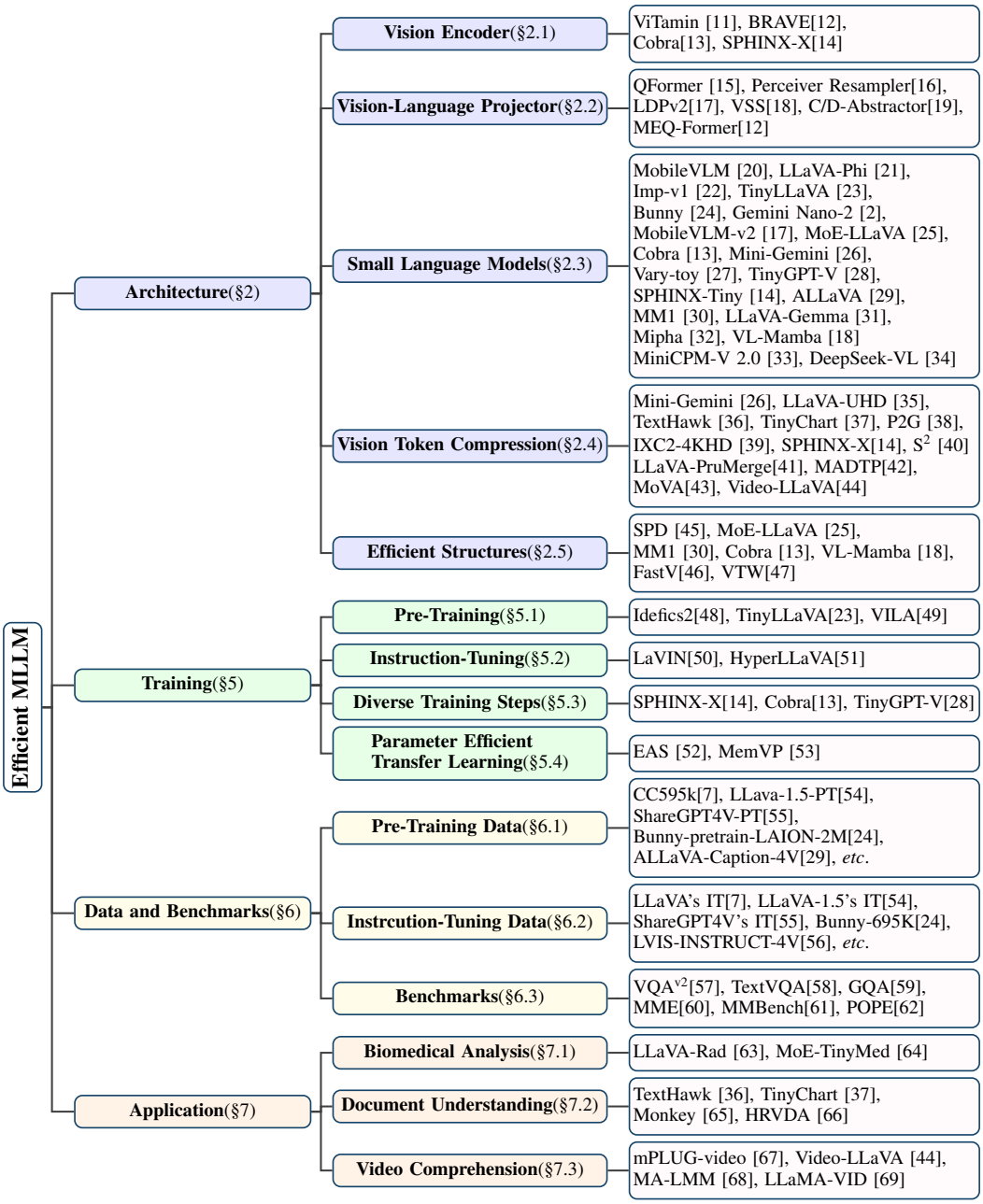


Figure 2: Organization of efficient multimodal large language models advancements.

- **Training** surveys the landscape of training methodologies that are pivotal in the development of efficient MLLMs. It addresses the challenges associated with the pre-training stage, instruction-tuning stage, and the overall training strategy for state-of-the-art results.
- **Data and Benchmarks** evaluates the efficiency of datasets and benchmarks used in the evaluation of multimodal language models. It assesses the trade-offs between dataset size, complexity, and computational cost, while advocating for the development of benchmarks that prioritize efficiency and relevance to real-world applications.
- **Application** investigates the practical implications of efficient MLLMs in various domains, emphasizing the balance between performance and computational cost. By addressing resource-intensive tasks such as high-resolution image understanding and medical

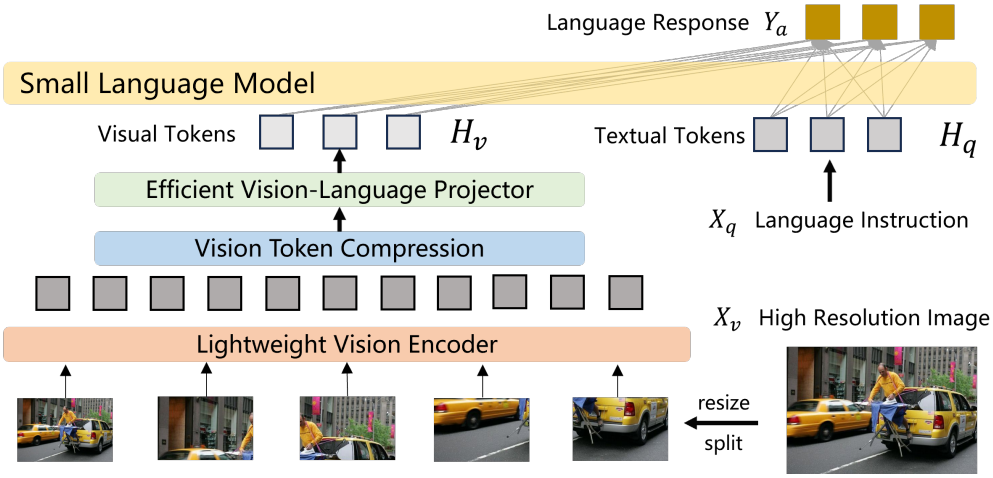


Figure 3: The architectures of efficient MLLMs.

question-answering, this section highlights the potential of efficient MLLMs to broaden their application scope and contribute to real-world problem-solving.

In summary, this survey delves into these research endeavors, exploring various strategies for making MLLMs more resource-efficient. We review the development history of efficient MLLMs, provide a taxonomy of the strategies for efficient MLLMs, and comprehensively compare the performance of existing efficient MLLMs.Through this exploration, we aspire to provide a comprehensive understanding of the current state-of-the-art, thereby illuminating the intricate nuances of this emerging field. Furthermore, this survey serves as a roadmap, highlighting potential avenues for future research, and fostering a deeper comprehension of the challenges and opportunities that lie ahead in the domain of efficient MLLMs. In addition to the survey, we have established a GitHub repository where we compile the papers featured in the survey, organizing them with the same taxonomy at https://github.com/lijiannuist/Efficient-Multimodal-LLMs-Survey. We will actively maintain it and incorporate new research as it emerges.

2 Architecture

Following the standard MLLM framework, efficient MLLMs can be divided into three main modules: a visual encoder g tasked with receiving and processing visual inputs, a pre-trained language model that manages the received multimodal signals and performs reasoning, and a visual-language projector P which functions as a bridge to align the two modalities. To enhance the efficiency of the general MLLMs, the primary optimization lies in handling high-resolution images, compressing vision tokens, implementing efficient structures, and utilizing compact language models, among other strategies. A diagram of the architecture is illustrated in Figure. 3. Table. 1 surveys a summary of the efficient MLLMs, which outlines the base LLM, the vision encoder, image resolution, and the projector used to connect vision and language. These efficient MLLMs include: MobileVLM [20], LLaVA-Phi [21], Imp-v1 [22], TinyLLaVA [23], Bunny [24], Gemini Nano-2 [2], MobileVLM-v2 [17], MoE-LLaVA-3.6B [25], Cobra [13], Mini-Gemini [26], Vary-toy [27], TinyGPT-V [28], SPHINX-Tiny [14], ALLaVA [29], MM1-3B [30], LLaVA-Gemma [31], Mipha-3B [32], VL-Mamba[18], MiniCPM-V2.0 [70], DeepSeek-VL [34], KarmaVLM [71], moondream2 [72]. In this section, we sequentially present a comprehensive overview of these three modules, along with other efficient components.

2.1 Vision Encoder

Taking the input image X_v as input, the vision encoder compresses the original image into more compact patch features Z_v, as represented by the following formula:

$$Z_v = g(X_v). \tag{1}$$

Model	Vision Encoder			LLM		Vision-LLM Projector
	Variants	Resolution	Parameter Size	Variants	Parameter Size	
MobileVLM [20]	CLIP ViT-L/14 [73]	336	0.3B	MobileLLaMA[20]	2.7B	LDP[20]
LLaVA-Phi [21]	CLIP ViT-L/14 [73]	336	0.3B	Phi-2[74]	2.7B	MLP
Imp-v1 [22]	SigLIP [75]	384	0.4B	Phi-2[74]	2.7B	-
TinyLLaVA [23]	SigLIP-SO [75]	384	0.4B	Phi-2[74]	2.7B	MLP
Bunny [24]	SigLIP-SO [75]	384	0.4B	Phi-2[74]	2.7B	MLP
MobileVLM-v2-3B [17]	CLIP ViT-L/14 [73]	336	0.3B	MobileLLaMA[17]	2.7B	LDPv2[17]
MoE-LLaVA-3.6B [25]	CLIP-Large [73]	384	-	Phi-2[74]	2.7B	MLP
Cobra [13]	DINOv2 [76] SigLIP-SO [75]	384	0.3B+0.4B	Mamba-2.8b-Zephyr[77]	2.8B	MLP
Mini-Gemini [26]	CLIP-Large [73]	336	-	Gemma[78]	2B	MLP
Vary-toy [27]	CLIP [73]	224	-	Qwen[79]	1.8B	-
TinyGPT-V [28]	EVA [80]	224/448	-	Phi-2[74]	2.7B	Q-Former [15]
SPHINX-Tiny [14]	DINOv2 [76] CLIP-ConvNeXt [81]	448	-	TinyLlama[82]	1.1B	-
ALLaVA-Longer [29]	CLIP-ViT-L/14 [73]	336	0.3B	Phi-2[74]	2.7B	-
MM1-3B-MoE-Chat [30]	CLIP_{DFN}-ViT-H [83]	378	-	-	3B*	C-Abstractor [19]
LLaVA-Gemma [31]	DinoV2 [76]	-	-	Gemma-2b-it[78]	2B	-
Mipha-3B [32]	SigLIP [75]	384	-	Phi-2[74]	2.7B	-
VL-Mamba [18]	SigLIP-SO [75]	384	-	Mamba-2.8B-Slimpj[77]	2.8B	VSS-L2[18]
MiniCPM-V 2.0[33]	SigLIP [75]	-	0.4B	MiniCPM[70]	2.4B	Perceiver Resampler [16]
DeepSeek-VL [34]	SigLIP-L [75]	384	0.4B	DeepSeek-LLM[84]	1.3B	MLP
KarmaVLM[71]	SigLIP-SO [75]	384	0.4B	Qwen1.5[79]	0.5B	-
moondream2[72]	SigLIP[75]	-	-	Phi-1.5[85]	1.3B	-
Bunny-v1.1-4B[24]	SigLIP[75]	1152†	-	Phi-3-Mini-4K[86]	3.8B	-

Table 1: The summary of 17 mainstream efficient MMLMs. * indicates activated parameters.†:High resolution support is achieved with S^2-Wrapper[40].

In line with mainstream MLLM practices, efficient MLLMs select pre-trained models that are semantically aligned with the text, represented by CLIP [73]. This approach facilitates better alignment between the feature spaces of visual and text inputs. Since the vision encoder constitutes a relatively minor portion of the MLLM parameters, the advantages of lightweight optimization are less pronounced compared to the language model. Therefore, efficient MLLMs generally continue to employ visual encoders that are widely used in large-scale MLLMs, as detailed in Table 1.

Multiple Vision Encoders BRAVE[12] in Figure. 4 performs an extensive ablation of various vision encoders with distinct inductive biases for tackling MLMM tasks. The results indicate that there isn't a single-encoder setup that consistently excels across different tasks, and encoders with diverse biases can yield surprisingly similar results. Presumably, incorporating multiple vision encoders contributes to capturing a wide range of visual representations, thereby enhancing the model's comprehension of visual data. Cobra[13] integrates DINOv2[76] and SigLIP[75] as its vision backbone, with the rationale that merging the low-level spatial features from DINOv2 and the semantic attributes offered by SigLIP will enhance performance on subsequent tasks. SPHINX-X[14] employs two vision encoders – DINOv2 and CLIP-ConvNeXt. Given that these models have been pre-trained via distinct learning methodologies (self-supervised versus weakly supervised) and network architectures (ViT versus CNN), they are naturally capable of offering the most complementary and sophisticated visual knowledge.

Lightweight Vision Encoder Vision Transformer architectures in real-world applications pose challenges due to hardware and environmental limitations, including processing power and computational capabilities. ViTamin [11] represents a lightweight vision model, specifically tailored for vision and language models. It commences with a convolutional stem, succeeded by Mobile Convolution Blocks in the first and second stages, and Transformer Blocks in the third stage. Remarkably, ViTamin-XL, with a modest count of 436M parameters, attains an 82.9% ImageNet zero-shot accuracy. This outperforms the 82.0% accuracy achieved by EVA-E [80], which operates with a parameter count ten times larger, at 4.4B. Simply replacing LLaVA's image encoder with ViTamin-L can establish new standards in various MLLM performance metrics.

2.2 Vision-Language Projector

The task of the vision-language projector is to map the visual patch embeddings Z_v into the text feature space:

$$H_v = P(Z_v), \tag{2}$$

where H_v denotes the projected visual embeddings. The aligned visual features are used as prompts and inputted into the language model along with the text embeddings. Vision-language projector

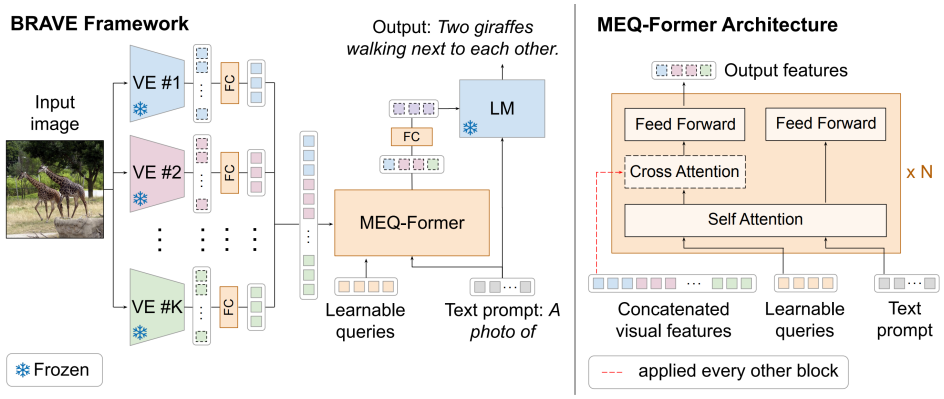


Figure 4: **BRAVE** [12] concatenates features from K different Vision Encoders in a sequence-wise manner. These concatenated features are then reduced by the MEQ-Former.

avoids the high cost of training an end-to-end multimodal model from scratch and effectively leverages the capabilities of pre-trained language and vision models.

MLP-based As outlined in [7, 54], the vision-language projector is typically realized using a straightforward, learnable Linear Projector or a Multi-Layer Perceptron (MLP), i.e., several linear projectors interleaved with non-linear activation functions, as illustrated in Table.1.

Attention-based BLIP2 [15] introduces Q-Former, a lightweight transformer, which employs a set of learnable query vectors to extract visual features from a frozen vision model. Perceiver Resampler, proposed by Flamingo[16], contemplates the use of learnable latent queries as Q in cross-attention, while image features are unfolded and concatenated with Q to serve as K and V in cross-attention. By this means, the transformer output at the corresponding positions of the learnable latent queries is taken as the aggregated representation of visual features, thereby standardizing variable-length video frame features into fixed-size features. MEQ-Former in BRAVE [12] designs a multi-encoder querying transformer to amalgamate features from multiple frozen vision encoders into a versatile representation that can be directly inputted into a frozen language model.

CNN-based MobileVLMv2[17] proposes LDPv2, a new projector consisting of three parts: feature transformation, token reduction, and positional information enhancement. By using point-wise convolution layers, average pooling, and a PEG module with a skip connection, LDPv2 achieves better efficiency, a 99.8% reduction in parameters, and slightly faster processing compared to the original LDP[20].

Mamba-based VL-Mamba[18] implements the 2D vision selective scanning(VSS) technique within its vision-language projector, facilitating the amalgamation of diverse learning methodologies. The VSS module primarily resolves the distinct processing approaches between one-dimensional sequential processing and two-dimensional non-causal visual information.

Hybrid Structure Honeybee [19] put forward two visual projectors, namely C-Abstractor and D-Abstractor, which adhere to two primary design principles: (i) providing adaptability in terms of the number of visual tokens, and (ii) efficiently maintaining the local context. C-Abstractor, or Convolutional Abstractor, focuses on proficiently modeling the local context by employing a convolutional architecture. This structure consists of L ResNet blocks, followed by adaptive average pooling and additional L ResNet blocks, which facilitate the abstraction of visual features to any squared number of visual tokens. Conversely, D-Abstractor, or Deformable attention-based Abstractor utilizes deformable attention, which maintains the local context through a 2-D coordinate-based sampling process, using reference points and sampling offsets.

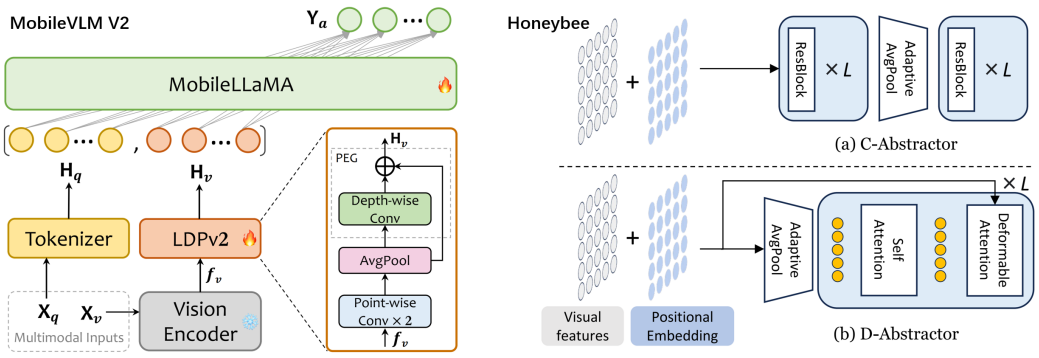


Figure 5: **MobileVLM v2** [17] and **Honeybee** [19] efficient vision-language projector.

2.3 Small Language Model

The pre-trained small language model(SLM) serves as the core component of MLLMs, endowing it with many outstanding capabilities, such as zero-shot generalization, instruction following, and in-context learning. The SLM accepts input sequences containing multiple modalities and outputs corresponding text sequences. A text tokenizer is typically bundled with the SLM, mapping text prompts X_q to the text tokens H_q. The text tokens H_q and the visual tokens H_v are concatenated as the input of the language model, which outputs the final response sequence Y_a in an autoregressive manner:

$$p(Y_a|H_v, H_q) = \prod_{i=1}^{L} p(y_i|H_v, H_q, y_{<i}), \tag{3}$$

where L denotes the length of Y_a. As the SLM contributes the vast majority of MLLM parameters, its selection is closely related to the lightweight nature of MLLM. In comparison to conventional MLLMs with parameter sizes ranging from 7 billion to tens of billions[87, 88], efficient MLLMs typically employ language models with less than 3 billion parameters, such as phi2-2.7B[74] by Microsoft and Gemma-2B[78] by Google. Phi-2 trained on special data recipes can match the performance of models 25 times larger trained on regular data. Phi-3-mini [86] can be easily deployed locally on a modern phone and achieves a quality that seems on-par with models such as Mixtral 8x7B [89] and GPT-3.5. In addition to utilizing pre-trained models, MobileVLM[20] downscales LLaMA[87] and trains from scratch using open-source datasets. The specific model scaling is illustrated in the Table.1 and Table.4.

2.4 Vision Token Compression

Initial research has underscored the potential of MLLMs across various tasks, including visual question answering and image captioning. However, MLLMs face considerable challenges in tasks necessitating intricate recognition, including crowd counting and OCR of small characters. A direct approach to address these challenges involves increasing the image resolution, practically, the number of visual tokens. This strategy, nonetheless, imposes a substantial computational burden on MLLMs, primarily due to the quadratic scaling of computational costs with the number of input tokens in the Transformer architecture. Motivated by this challenge, vision token compression, aimed to reduce the prohibitive computation budget caused by numerous tokens, has become an essential aspect of efficient MLLMs. We will explore this topic through several key techniques, including multi-view input, token processing, multi-scale information fusion, vision expert agents and video-specific methods.

Multi-view Input Directly employing high-resolution vision encoders for fine-grained perception is prohibitively costly and does not align with practical usage requirements. Therefore, to utilize low-resolution vision encoders while enabling MLLM to perceive detailed information, a common approach is to input multi-view HR images, *i.e.*, a global view: low-resolution images obtained through resizing, and a local view: image patches derived from splitting. For example,

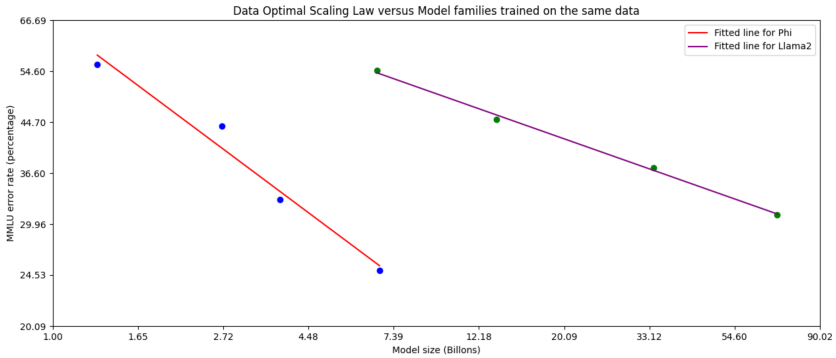


Figure 6: Comparision of **Phi** [86] (from left to right: phi-1.5, phi-2, phi-3-mini, phi-3-small) versus **Llama-2** [91] family of models(7B, 13B, 34B, 70B) that were trained on the same fixed data.

LLaVA-UHD [35] proposes an image modularization strategy that divides native-resolution images into smaller variable-sized slices for efficient and extensible encoding. Inaddition, InternLM-XComposer2-4KHD [90] introduces a strategy that dynamically adjusts resolution with an automatic layout arrangement, which not only maintains the original aspect ratios of images but also adaptively alters patch layouts and counts, thereby enhancing the efficiency of image information extraction. By implementing an adaptive input strategy for images of varying resolutions, a balance between perceptual capability and efficiency can be achieved.

Token Processing Techniques designed to process lengthy visual token squence are critical in efficient MLLMs as they address the dual challenges of preserving fine-grained details and reducing computational complexity. LLaVA-UHD [35] presents a novel approach to manage the computational burden associated with high-resolution images. It puts forward two key components: (1) a compression module that further condenses image tokens from visual encoders, significantly reducing the computational load, and (2) a spatial schema to organize slice tokens for LLMs. Notably, LLaVA-UHD demonstrates its efficiency by supporting 6 times larger resolution images using only 94% of the inference computation compared to previous models. Furthermore, the model can be efficiently trained in academic settings, completing the process within 23 hours on 8 A100 GPUs. LLaVA-PruMerge[41] and MADTP [42] propose an adaptive visual token reduction approach that significantly decreases the number of visual tokens while preserving comparable model performance. TinyChart [37] and TextHawk [36] focus on document-oriented tasks, with the former adopting the Vision Token Merging module and the latter introducing the ReSampling and ReArrangement module. These modules can enhance fine-grained visual perception and information compression capabilities.

Multi-Scale Information Fusion Utilizing multi-scale image information is indeed crucial for visual feature extraction. This approach allows the model to capture both the fine-grained details present in smaller scales and the broader context available in larger scales. Mini-Gemini [26] comprises twin encoders, one for high-resolution images and the other for low-resolution visual embedding. It proposes Patch Info Mining, which uses low-resolution visual embeddings as queries to retrieve relevant visual cues from high-resolution candidates through cross-attention. Scaling on Scales (S^2) [40] demonstrated that a multi-scale smaller model has comparable learning capacity to a larger model, and pre-training smaller models with S^2 can match or even exceed the advantage of larger models on MLLM benchmarks while being more compute-efficient. After splitting the large image into small sub-images, S^2-wrapper processes individual sub-images instead of using window attention, which allows using a pre-trained model that does not support window attention and avoids training additional parameters from scratch.It then interpolates the large feature map into the regular size, making sure the number of visual tokens stays acceptable.

Vision Expert Agents Most MLLMs, due to their non-lossless image tokenization, struggle to fully capture the intricate details of text and objects. Leveraging vision expert agents is a solution to the problem of a single vision encoder's limited generalization ability on detail-abundant content. P2G [38] employs expert agents for real-time grounding, enabling efficient and purposeful reasoning

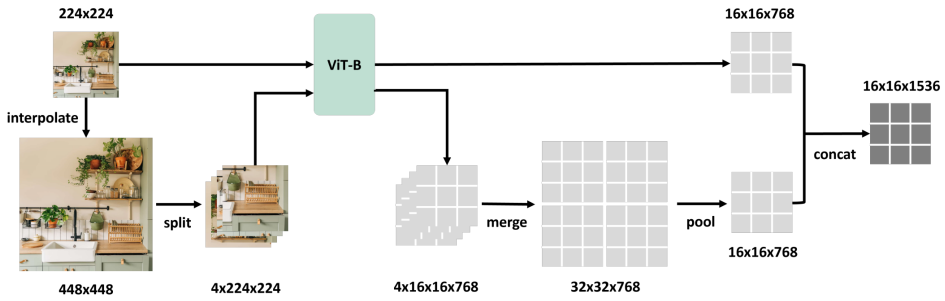


Figure 7: **S^2-Wrapper** [40] is a simple mechanism that extends any pre-trained vision model to multiple image scales in a parameter-free manner.

through multimodal prompting. This innovative framework facilitates plug-and-play grounding of reasoning in high-resolution scenarios that are rich in natural visuals and text. It achieves this by leveraging agents to enhance both textual and visual grounding and perception, like OCR Agent (text) or Grounding Agent (image). MoVA[43] addresses the issue of diminished generalization ability of an individual vision encoder across various contents by introducing an expert routing strategy. This approach enables the flexible and effective utilization of representations from multiple task-specific vision experts, thereby enhancing the generalization capabilities.

Video-Specific Methods Video understanding also requires processing a large number of frames, which can pose a significant computational challenge within the context window of LLMs. Elysium [92] provides a trade-off between performance and visual token consumption, where T-Selector is introduced as a visual token compression network to enable LLMs to distinguish individual frames while reducing visual token use. VideoLLaVA [44], building upon LanguageBind [93], unifies visual representation into the language feature space to advance foundational LLMs towards a unified language-vision LLM without incurring a large computational burden.

2.5 Efficient Structures

Efficient structures primarily explore three directions: Mixture-of-Experts, Mamba and Inference Acceleration.

Mixture of Experts MoE enhances model capacity by modulating the total count of model parameters while maintaining the activated parameters unchanged, hence, not significantly compromising the inference speed. MoE-LLaVA[25] presents an MoE-based sparse MLLM framework that effectively increases the number of parameters without compromising computational efficiency. Furthermore, it introduces MoE-Tuning, a three-stage training strategy designed to adapt MoE [89] to MLLMs and prevent model degradation caused by sparsity. MM1[30] designs two variants of MoE models. The first is a 3B-MoE model that employs 64 experts and substitutes a dense layer with a sparse one every two layers. The second is a 7B-MoE model that utilizes 32 experts and substitutes a dense layer with a sparse one every four layers.

Mamba Cobra [13] incorporates the efficient Mamba [77] language model into the vision modality and explores different modal fusion schemes to develop an effective multi-modal Mamba. Experiments show that it not only achieves competitive performance with state-of-the-art efficient methods but also boasts faster speeds due to its linear sequential modeling.It also excels in overcoming visual illusions and spatial relationship judgments in closed-set challenging prediction benchmarks and achieves performance comparable to LLaVA while using only 43% of the parameters. VL-Mamba[18] substitutes the Transformer-based backbone language model with the pre-trained Mamba language model. It explores how to effectively implement the 2D vision selective scan mechanism for multimodal learning and the combinations of different vision encoders and pre-trained Mamba language model variants.

Inference Acceleration SPD[45] proposes the speculative decoding with a language-only model to improve inference efficiency. By employing a language-only model as a draft model for specu-

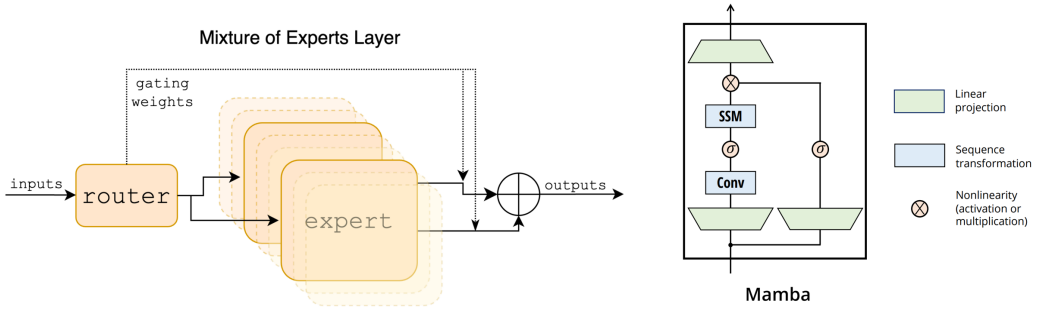


Figure 8: Structure of **MOE** [89](left) and **Mamba** [77](right).

lative decoding, the need for image tokens and their associated processing components is bypassed. FastV [46] finds that most image tokens receive inefficient attention after the second decoder layer and achieve computation reduction by eliminating redundant visual tokens during the inference stage without sacrificing performance. VTW [47] asserts that visual tokens are not essential in the deeper layers of MLLM. It strategically removes all of them at a specific layer, allowing only text tokens to participate in the subsequent layers. This approach by VTW can reduce computational overhead by more than 40% across a variety of multimodal tasks, without compromising performance.

3 Efficient Vision

Vision Transformer (ViT) [94] architectures have gained significant popularity and are widely used in computer vision applications. However, as ViT models have grown in size, the number of trainable parameters and operations has also increased, impacting their deployment and performance. Additionally, the computational and memory cost of self-attention grows quadratically with image resolution. Referring to the paper [95], this survey aims to explore the most efficient vision encoding methodologies that may be used for efficient MLLMs.

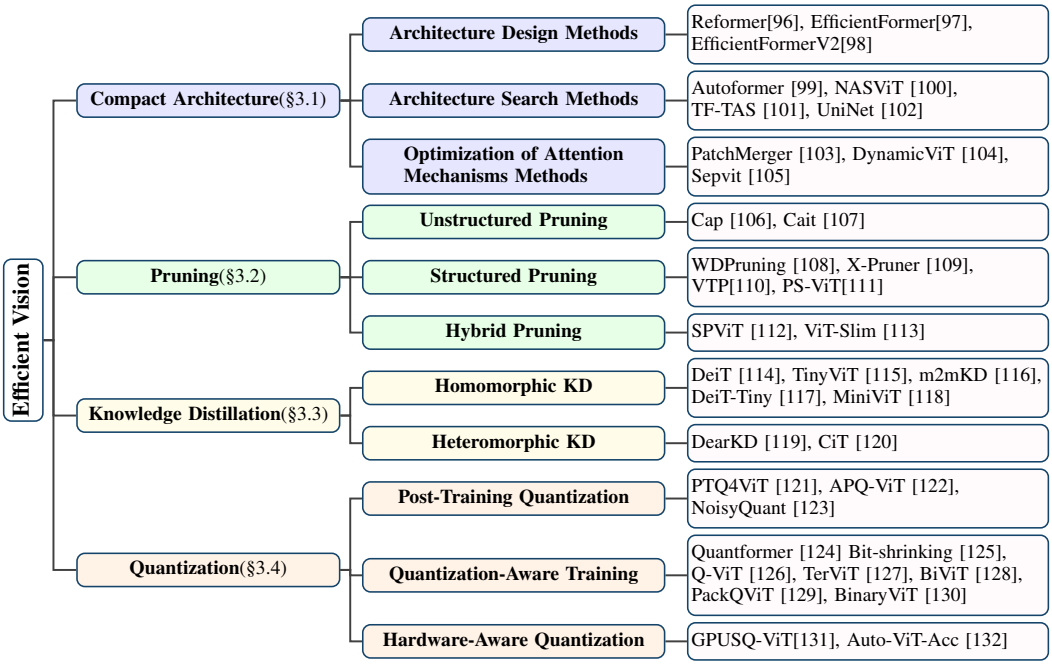


Figure 9: Organization of efficient vision advancements.

3.1 Compact Architecture

Compact Architecture refers to the design of lightweight and efficient models while maintaining high performance in downstream tasks. It encompasses various strategies and methodologies to reduce model size, computational complexity, and memory footprint without compromising performance. These strategies can be broadly categorized into three categories, 1) Architecture Design Methods, 2) Architecture Search Methods, and 3) Optimization of Attention Mechanisms Methods.

Architecture Design Methods involve creating new architectures [133] or adjusting existing ones [134] to achieve compactness without sacrificing performance. For example, Reformer [96] introduced locality-sensitive hashing in attention mechanisms to reduce complexity, while also employing reversible residual layers to store activations more efficiently. Furthermore, EfficientFormer [97] analyzed ViT-based model architectures and operators, introducing a dimension-consistent pure transformer paradigm and employing latency-driven slimming to produce optimized models. Additionally, EfficientFormerV2 [98] proposed a supernet with low latency and high parameter efficiency.

Architecture Search Methods involve employing neural architecture search algorithms [113] to explore and discover compact architectures tailored to specific tasks or constraints. For instance, Autoformer [99] intertwined weights within layers, enabling thorough training of thousands of subnets. NASViT [100] introduced a gradient projection algorithm, switchable layer scaling, and streamlined data augmentation, enhancing convergence and performance. Additionally, TF-TAS [101] investigated training-free architecture search methods, proposing an efficient scheme. UniNet [102] introduced context-aware down-sampling modules improving information accommodation by transformer and MLP operators.

Optimization of Attention Mechanisms Methods focus on reducing computational complexity by introducing adaptive attention, learning sparse attention patterns, and dynamically adjusting attention mechanisms. Fayyaz *et al.* [135] implemented adaptive attention by scoring and adaptively sampling significant tokens. PatchMerger [103] extracted global information among regional tokens and exchanged local self-attention with information among regional tokens via self-attention. DynamicViT [104] proposed an attention masking strategy to differentiably prune tokens by blocking interactions with other tokens. Additionally, Sepvit [105] conducted local-global information interaction within and across windows using depthwise separable self-attention. These methods collectively optimize attention mechanisms, enhancing computational efficiency and performance.

3.2 Pruning

Pruning involves removing less essential weights from vision transformer models, typically categorized as unstructured pruning, structured pruning, and hybrid pruning techniques.

Unstructured Pruning focuses on eliminating individual weights without considering their structural arrangement within the model. Rao *et al*. [104] introduced a dynamic token sparsification framework for progressive and adaptive pruning of redundant tokens based on input, integrating a lightweight prediction module to estimate token importance scores and employing an attention masking strategy to differentiate token interactions and optimize the prediction module in an end-to-end fashion. Cap [106] proposed a novel theoretically-grounded pruner capable of accurately and efficiently handling intricate weight correlations during pruning, alongside an effective fine-tuning procedure for post-compression recovery. Cait [107] introduced asymmetric token merging to integrate neighboring tokens efficiently while preserving the spatial structure, paired with consistent dynamic channel pruning for uniform pruning of unimportant channels in Vision Transformers, enhancing model compression.

Structured Pruning aims to remove structural components, such as attention heads or layers based on predefined criteria. For example, WDPruning [108] employed a binary mask to discern insignificant parameters based on their magnitudes. Additionally, Yu *et al*. [136] presented a unified framework integrating pruning to generate compact transformers. X-Pruner [109] utilizes an end-to-end learned explainability-aware mask to measure each unit's contribution to predicting target classes and adaptively searches layer-wise thresholds to preserve the most informative unit while

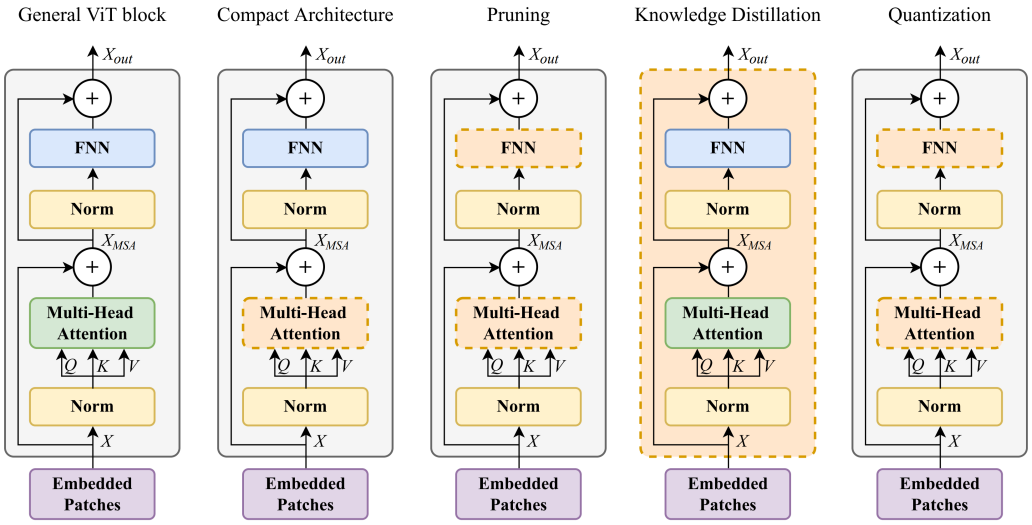


Figure 10: Efficient vision transformer techniques in [138]. The dashed orange block highlights the component on which each optimization technique mainly focuses.

determining the pruning rate. Additionally, VTP [110] reduces embedding dimensions through the integration of control coefficients, concurrently removing neurons with negligible coefficients. Tang *et al.* [111] eliminate redundant patches by first identifying effective patches in the last layer and then leveraging them to guide the selection process of previous layers, where patches with minimal impact on the final output feature are subsequently discarded.

Hybrid Pruning , such as [137], investigates both unstructured and structured sparsity, introducing a first-order importance approximation approach for attention head removal. SPViT [112] develops a dynamic attention-based multi-head token selector for adaptive instance-wise token selection, alongside a soft pruning technique consolidating less informative tokens into package tokens rather than discarding them. ViT-Slim [113] utilizes a learnable and unified sparsity constraint with pre-defined factors to represent global importance within the continuous search space across various dimensions.

3.3 Knowledge Distillation

Knowledge distillation is a technique in which a smaller model learns from a larger, more complex model to replicate its performance, enabling efficient deployment while maintaining predictive accuracy [139]. Knowledge distillation (KD) techniques for Vision Transformers (ViTs) can be categorized into two main types: 1) homomorphic KDs and 2) heteromorphic KDs.

Homomorphic KDs can further classified into logit-level [114, 115], patch-level [117], module-level [116], and feature-level KDs [118]. For logit-level methods, in DeiT [114], a distillation token is incorporated into the self-attention module to emulate the class label inferred by the teacher model, facilitating interaction between the student attention and layers, thus enabling the learning of hard labels during back-propagation. TinyViT [115] applies distillation during pretraining, where logits from large teacher models are pre-stored in the hardware, enabling memory and computational efficiency when transferring knowledge to scaled-down student transformers. Patch-level techniques like DeiT-Tiny [117] train a small student model to match a pre-trained teacher model on patch-level structures, then optimize with a decomposed manifold matching loss for reduced computational costs. Module-level methods involve segregating teacher modules from a pre-trained unified model, and student modules from a modular model. In m2mKD [116], these modules are combined with a shared meta-model, allowing the student module to emulate the behavior of the teacher module. Feature-level KD methods, as demonstrated by MiniViT [118], combine the weights of consecutive transformer blocks. This entails sharing weights across layers while introducing transformations to

enhance diversity. Additionally, weight distillation over self-attention is utilized to transfer knowledge from large-scale ViT models to compact models with multiplexed weights.

Heteromorphic KDs involves transferring knowledge between models with differing architectures. For example, DearKD [119] proposes a novel two-stage framework, DearKD, departing from traditional methods for ViT architectures. In the first stage, they use a vanilla KD strategy to transfer CNN features to the ViT student model, representing a heteromorphic transfer. In the subsequent phase, if real samples are limited, they introduce a boundary-preserving intra-divergence loss to enhance the process. Similarly, CiT [120] proposes a heteromorphic KD strategy, where knowledge is transferred from diverse models, such as a CNN and an involution neural network, resulting in improved performance for the ViT student model.

3.4 Quantization

ViT quantization is the process of reducing the precision of numerical representations in ViT models, typically transitioning from floating-point to fixed-point arithmetic [140]. This reduction in precision aims to decrease memory usage, computational complexity, and energy consumption while preserving model accuracy to an acceptable level. Current research can be mainly categorized into post-training quantization, quantization-aware training, and hardware-aware quantization.

Post-Training Quantization (PTQ) compresses trained ViT models by converting their parameters from high-precision floating-point numbers to lower-precision fixed-point numbers, such as 8-bit integers. For example, Liu *et al.* [141] introduced a ranking loss method to identify optimal low-bit quantization intervals for weights and inputs, ensuring the functionality of the attention mechanism. They also conducted an analysis to understand the relationship between quantization loss in different layers and feature diversity, exploring a mixed-precision quantization approach leveraging the nuclear norm of each attention map and output feature. Additionally, PTQ4ViT [121] introduced the twin uniform quantization method to minimize quantization error on activation values following softmax and GELU functions, incorporating a Hessian-guided metric to enhance calibration accuracy. APQ-ViT [122] proposed a unified Bottom-elimination Blockwise Calibration scheme to optimize the calibration metric, prioritizing crucial quantization errors and designing a Matthew-effect Preserving Quantization for Softmax to maintain the power-law character and attention mechanism functionality. NoisyQuant [123] proposes to add a fixed Uniform noisy bias to quantized values, the quantization error is significantly reduced under certain conditions. This technique successfully modifies heavy-tailed activation distributions to fit a given quantizer.

Quantization-Aware Training (QAT) integrates quantization into the training cycle. This integration is particularly advantageous when scaling down to ultra-low bit precision, such as 4 bits or lower, where PTQ struggles with significant performance loss. For example, Quantformer [124] leverages entropy information to maintain consistency in self-attention ranks and introduces a differentiable search mechanism to optimally group patch feature dimensions, reducing rounding and clipping inaccuracies. Q-ViT [126] incorporates a distillation token and Information Rectification Module (IRM) to counteract altered distributions in quantized attention modules. TerViT [127] and Bit-shrinking [125] progressively reduce model bit-width while regulating sharpness to maintain accuracy throughout quantization. PackQViT [129] mitigates outlier effects during quantization. BiViT [128] introduces Softmax-aware Binarization to adjust the binarization process, minimizing errors in binarizing softmax attention values. Xiao *et al.* [142] integrated a gradient regularization scheme to curb weight oscillation during binarization training and introduced an activation shift module to reduce information distortion in activations. Additionally, BinaryViT [130] integrates essential architectural elements from CNNs into a pure ViT framework, enhancing its capabilities.

Hardware-Aware Quantization optimizes the quantization process of neural network models for specific hardware platforms (*e.g.*, GPUs [131], FPGA [132]). It adjusts precision levels and quantization strategies to maximize performance and energy efficiency during inference. For example, Yu *et al.* [131] propose a compression scheme utilizing GPU-friendly 2:4 fine-grained structured sparsity and quantization. They prune a dense model into a sparse one using 2:4 structured pruning, leveraging GPU acceleration. Then, they quantize the sparse model into fixed-point representation through sparse-distillation-aware quantization-aware training, exploiting GPU speedup. Throughout the process, they employ mixed-strategy knowledge distillation, enabling support for supervised and

unsupervised learning styles. Auto-ViT-Acc [132] proposed a framework designed for quantizing ViT architectures to run inference on FPGA-powered devices. They apply the quantization function from a prior study specifically to the FNN module within the attention block, aiming to optimize FPGA resource utilization and accelerate inference.

4 Efficient LLMs

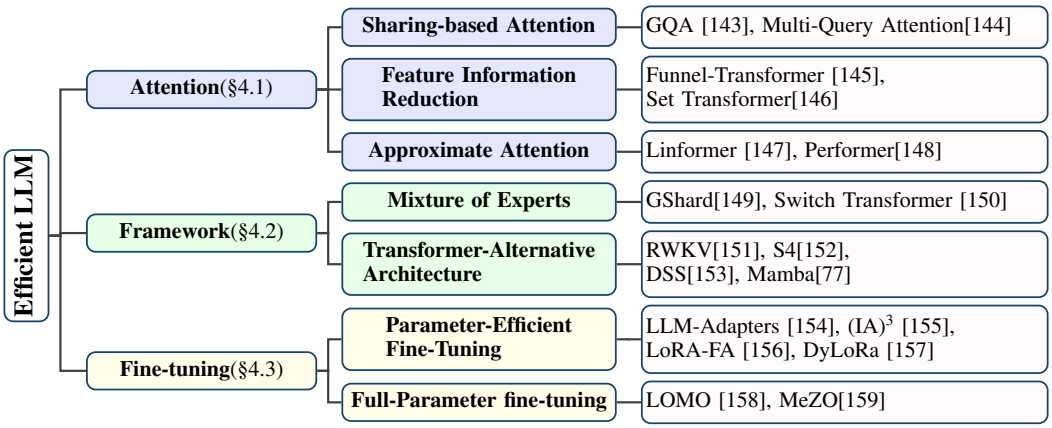


Figure 11: Organization of efficient large language models advancements.

Occupying a significant majority of the parameter volume in MLLMs, LLM serves as a crucial entry point for enhancing the efficiency of MLLMs. In this section, similar to the survey paper [160], we provide a brief overview of the research progress in efficient LLMs, offering inspiration for the development of Efficient MLLMs.

4.1 Attention

In the standard self-attention mechanism, the time complexity is $O(n^2)$, where n is the sequence length. This quadratic complexity arises due to the pairwise interactions between all input tokens, which can lead to scalability issues, especially when dealing with long sequences in LLMs. To tackle this, researchers have developed techniques to expedite attention mechanisms and reduce time complexity, such as sharing-based attention, feature information reduction, kernelization or low-rank, fixed and learnable pattern strategies, and hardware-assisted attention.

Sharing-based Attention Sharing-based Attention aims to expedite attention computation during inference by by sharing computation resources across multiple Key-Value heads. For example, Llama-2 [91] incorporates a technique called grouped-query attention (GQA) [143] to optimize memory bandwidth during the autoregressive decoding. GQA is a Sharing-based Attention technique that seeks to achieve a balance between performance and efficiency, positioned between multi-head attention and multi-query attention [144] mechanisms. In multi-head attention, each head utilizes a distinct set of linear transformation parameters for queries, keys, and values. Conversely, multi-query attention shares a single set of key-value heads across all queries. GQA partitions all query heads into several groups, with each group's query heads sharing a common key-value head, thereby establishing a rigorous equilibrium between effectiveness and computational cost.

Feature Information Reduction Feature Information Reduction, as evidenced by models such as Funnel-Transformer[145] and Set Transformer[146], addresses the crucial need for computational efficiency in attention mechanisms, specifically by reducing the dimensionality or quantity of input features while preserving the essential information embedded within the data. A key motivation behind this strategy stems from the potential redundancy in maintaining full-length hidden representations across all layers in Transformer models. Funnel-Transformer [145] tackles this issue by progressively reducing the sequence size of hidden representations in self-attention models, such as

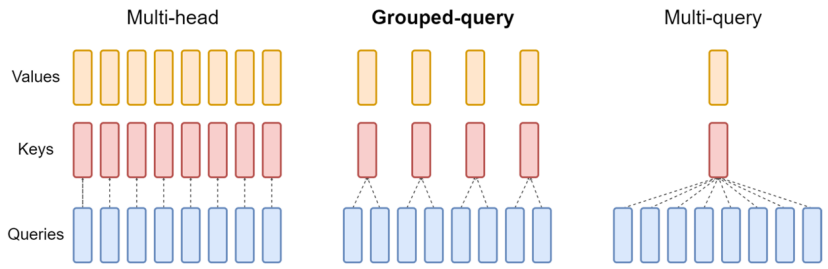


Figure 12: In GQA[59], a single set of key and value heads is allocated for each group of query heads, providing a balance between multi-head and multi-query attention mechanisms.

sequence length. This reduction not only decreases computational complexity and memory usage but also frees up resources that can be allocated toward building deeper or wider models.

Approximate Attention Approximate Attention facilitates models to efficiently focus on task-relevant information when processing long texts. Two pivotal concepts within Approximate Attention are Kernelization and Low-Rank. Kernelization, e.g. [148], involves transforming a problem into a kernel-based framework with the goal of converting the original issue into a more manageable one in a higher-dimensional space. Kernelization is primarily employed to map text sequences into a high-dimensional space, where task-related information can be more readily captured. In this new space, each word in the text sequence is represented as a high-dimensional vector, and the distances between these vectors serve to measure their similarities. Low-Rank [147] aims to decompose a high-dimensional matrix into the product of two lower-dimensional matrices. Consequently, by calculating the inverses of these two lower-dimensional matrices, an approximate inverse of the attention matrix can be obtained, thereby significantly reducing computational complexity.

4.2 Framework

Mixture of Experts The core idea behind MoE [89] is to decompose a large-scale model into several smaller models, each of which focuses on learning a specific part of the input data. During the training process, each expert is assigned a weight that determines its importance within the overall model. During the inference phase, given an input, all experts are ranked, and the most relevant ones are selected for computation. This approach considerably reduces the amount of computation, as only a subset of experts is involved in the calculation.By distributing computational tasks among different experts, MoE achieves more efficient utilization of computational resources during both training and inference phases. In MoE, each expert has its own set of parameters; however, these parameters are shared during the training process. This parameter-sharing strategy reduces the overall number of parameters in the model, consequently lowering storage and computational costs. GShard [149] is a module composed of a set of lightweight annotation APIs and XLA compiler extensions, which offers an elegant way to express various parallel computation patterns while making minimal changes to existing model code. It enables us to scale multi-lingual neural machine translation Transformer models with sparse gated mixtures of experts to over 600 billion parameters using automatic sharding. Switch Transformer [150] replaces the feedforward network (FFN) layer in the standard Transformer with a MoE routing layer, where each expert operates independently on the tokens in the sequence. Its training speed is four times faster than Google's previously developed largest model, T5-XXL, under the same computational resources. The proposed training techniques have eliminated instability during the training process, demonstrating that large sparse models can also be trained in a low-precision format, such as bfloat16.

Transformer-Alternative Structures Although the Transformer is the dominant architecture in current large-scale language models, models like RWKV [151] and Mamba [77] have emerged as popular solutions for achieving heightened efficiency and processing lengthy texts. These innovative models have demonstrated attributes similar to transformers, including the ability to handle long-range dependencies and parallel processing.RWKV model leverages a linear attention mechanism, enabling us to formulate the model as either a Transformer or a Recurrent Neural Network (RNN).

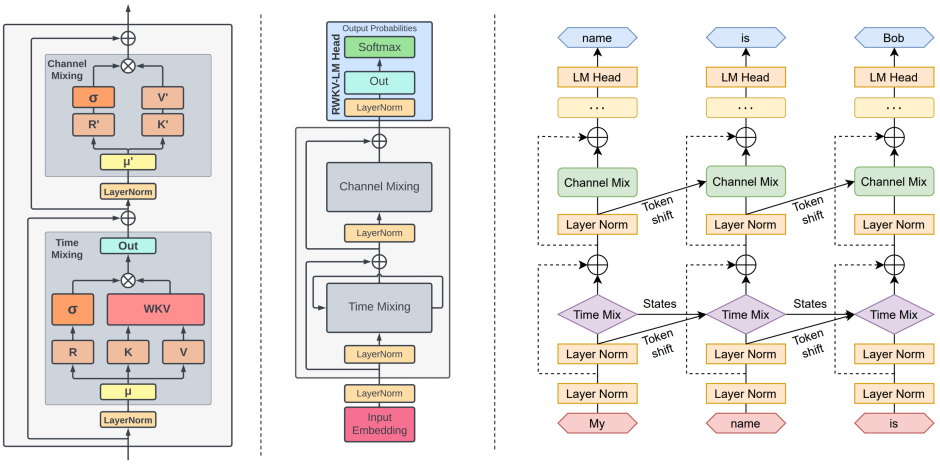


Figure 13: The elements(left) block(middle) and architecture(right) in RWKV [151].

This approach parallelizes computations during training and maintains constant computational and memory complexity during inference.

State Space Models (SSMs) [152] can be formulated as a type of RNN for efficient autoregressive inference and have emerged as a promising alternative to attention mechanisms, offering near-linear computational complexity compared to the quadratic complexity of attention. SSMs are formulated as $x'(t) = Ax(t) + Bu(t)$, $y(t) = Cx(t) + Du(t)$, mapping a single-dimension input signal $u(t)$ to an N-dimension latent state $x(t)$ before projecting it to a single-dimension output signal $y(t)$, with A, B, C, and D being parameters learned by gradient descent [152]. Several techniques have been proposed to enhance SSMs, such as the Structured State Space sequence model (S4) [152], which refines SSMs by conditioning matrix A with a low-rank correction, and the Diagonal State Space (DSS) model [153], which proposes fully diagonal parameterization of state spaces for greater efficiency. H3 stacks two SSMs to interact with their output and input projection, bridging the gap between SSMs and attention while adapting to modern hardware. Mamba [77], a selective state space model, has been introduced as a strong competitor to the Transformer architecture in large language models. Mamba incorporates a selection mechanism to eliminate irrelevant data and develops a hardware-aware parallel algorithm for recurrent operation. This results in competitive performance compared to LLMs of the same capacity, with faster inference speeds that scale linearly with time and constant memory usage. In conclusion, State Space Models offer significant potential as an alternative to attention mechanisms by providing near-linear computational complexity and effectively capturing long-range dependencies. With continuous advancements and refinements, SSMs are poised to become an influential approach in the field of deep learning and sequence processing.

4.3 Fine-Tuning

Fine-tuning, as the primary stage for adapting LLMs to downstream tasks and training MLLLMs to follow visual instructions, plays a crucial role in enhancing the efficiency of LLMs.

Parameter-Efficient Fine-Tuning Parameter-Efficient Fine-Tuning (PEFT) is an approach that aims to achieve high performance with fewer parameters in Large Language Models (LLMs). Techniques such as adapter-based tuning and low-rank adaptation provide effective solutions to mitigate the computational and memory challenges associated with fine-tuning LLMs while maintaining their expressiveness and generalization capabilities. Adapter-based tuning introduces lightweight adapter modules into the pre-trained model's architecture. These adapter modules, typically composed of feed-forward neural networks with a small number of parameters, are inserted between the layers of the original model. During fine-tuning, only the adapter parameters are updated, while the pre-trained model's parameters remain fixed. This method significantly reduces the number of trainable parameters, leading to faster training and inference times without compromising the model's performance. LLM-Adapters [154] presents a framework for integrating various adapters into large language models, enabling parameter-efficient fine-tuning for diverse tasks. This framework en-

compasses state-of-the-art openly accessible large language models and a wide range of widely-used adapters.$(IA)^3$ [155] introduces a novel Parameter-Efficient Fine-Tuning method, Infused Adapters by Inhibiting and Amplifying Inner Activations, which learns vectors to weight model parameters through multiplication with activations, enabling robust few-shot performance and task mixing within batches during inference without manual model structure adjustments. Low-rank adaptation [161] employs matrix factorization techniques to reduce the number of parameters in the model. By decomposing the original weight matrices into lower-rank matrices, low-rank adaptation captures the most significant components of the model's representations while discarding less important information. This results in a more compact model with a reduced number of parameters, which can be fine-tuned more efficiently.In LoRA-FA [156], a variant of LoRA, the first low-rank matrix is frozen after initialization and used as a random projection, while the other is trained. This leads to a reduction in the number of parameters by half, while maintaining a performance comparable to the conventional LoRA technique.DyLoRa [157] introduces a dynamic low-rank adaptation technique that enables the training of LoRA blocks for a range of ranks instead of a single rank, which is achieved by sorting the representations learned by the adapter modules during training across different ranks.

Full-Parameter fine-tuning Full-parameter fine-tuning is an approach in which all the parameters of a pre-trained model are updated during the fine-tuning process. This method aims to achieve optimal performance on a specific downstream task by leveraging the entire capacity of the pre-trained model. While full-parameter fine-tuning often leads to state-of-the-art results and improved task-specific performance, it comes with higher resource requirements in terms of computational power and memory consumption. In an effort to lessen the burden associated with training, numerous studies have concentrated on enhancing memory efficiency during full-parameter fine-tuning. This strategic approach has effectively diminished the obstacles that once hindered progress in this field of research. LOMO [158] introduces a Low-Memory Optimization technique derived from Stochastic Gradient Descent (SGD) to reduce memory consumption. Typically, the ADAM optimizer is employed; however, the optimizer states in this approach occupy a significant amount of memory. By utilizing the modified SGD-based LOMO, memory usage can be reduced. While SGD itself faces three challenges, these issues tend to resolve themselves during model fine-tuning. The specific modification involves updating the parameters within the gradient computation rather than after an entire layer. MeZO[159] proposes an optimizer that computes gradients using merely two forward passes, enabling the fine-tuning of LLMs with a memory footprint equivalent to that of inference. With a GPU memory requirement of 55GB, it allows for the comprehensive fine-tuning of a 30B parameter model.

5 Training

The training process of efficient MLLMs is a critical aspect that determines their performance on downstream tasks and their ability to handle diverse modalities. In this section, we provide an overview of various training methodologies, including pre-training, instruction-tuning, diverse training steps, and parameter-efficient transfer learning strategies. These approaches aim to optimize the alignment between different modalities, fine-tune the models on specific tasks, and minimize the computational and parameter costs associated with the transfer learning process. Figure.14 presents a schematic representation of the different training stages involved in the development of efficient MLLMs. In the following subsections, we delve deeper into each of these aspects and discuss their significance in the context of efficient MLLMs.

5.1 Pre-Training

In the pre-training stage, the primary focus is on aligning different modalities in the embedding space, enabling the language model to accept inputs from various modalities. This phase of training mainly involves large-scale text-paired data, predominantly in the form of image-caption pairs. An image-caption pair (X, Y) is typically expanded into a single-turn conversation $(X_{instruct}, X_a)$, where $X_{instruct}$ contains an image X_v and a randomly sampled question X_q from a set of instructions asking the assistant to briefly describe the image, and X_a is the original image description. Given such a conversation, the model is trained to autoregressively predict the image description. Consequently, we can compute the probability of predicting X_a conditioned by X_v and optimize it

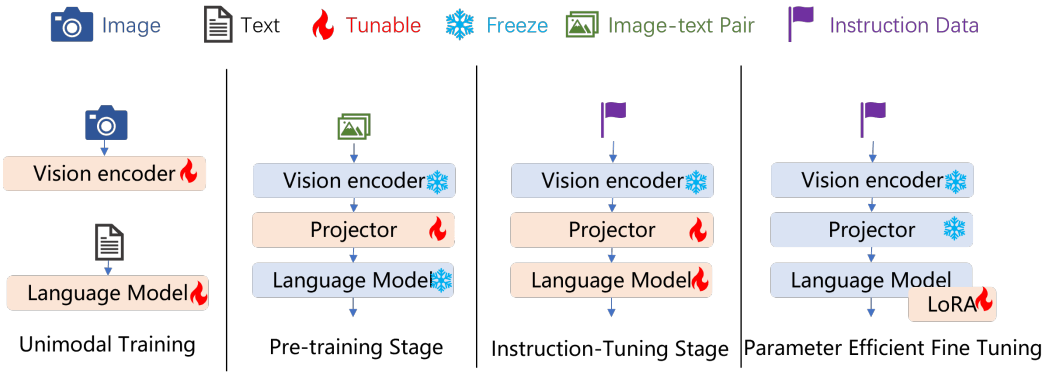


Figure 14: Training stages of efficient MLLMs.

using a standard cross-entropy loss function:

$$\max_{\theta} \sum_{i=1}^{L} \log p_{\theta}(x_i | X_v, X_{instruct}, X_{a,<i}), \tag{4}$$

where L is the length of X_a and θ denotes the trainable parameters. In order to better align different modalities of knowledge and avoid catastrophic forgetting during the pre-training stage, θ typically includes only a learnable modality interface, *i.e.*, a vision-language projector.

Which part to unfreeze? Considering that only training the connector may not well align the vision and text information when using SLMs, TinyLlava[23] also opt to partially freeze pre-trained modules (i.e. vision encoder and SLM) to activate more parameters for learning alignment. VILA[49] reveals that updating the base LLM throughout the pre-training stage is essential to inheriting some of the appealing LLM properties like in-context learning. ShareGPT4V[55] found that unfreezing more parameters, particularly in the latter half of the vision encoder's layers, proves beneficial when learning larger and more diverse datasets, showing the choice of training recipe is closely related to the quality of the data.

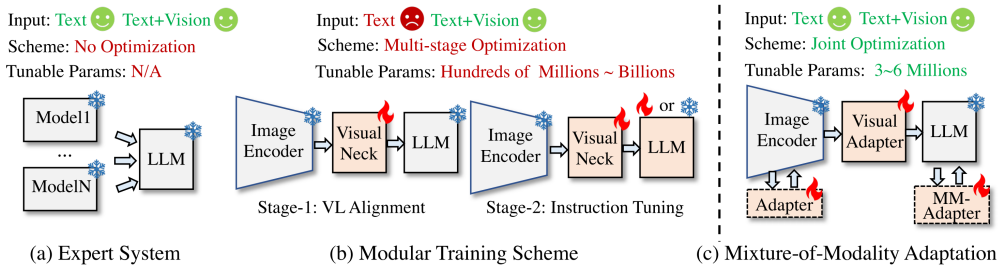
Multi-stage pre-training To maximize compute efficiency, Idefics2 [48] decomposes the pre-training in two stages. In the first stage, it limits the max image resolution to 384 pixels and use a large global batch size. In the second stage, PDF documents are introduced to increase image resolution to a maximum of 980 pixels for the text to be legible.

5.2 Instruction-Tuning

Instruction-tuning (IT) is a crucial aspect of efficient MLLMs, which aims to fine-tune the models on specific tasks by leveraging task-specific instructions. This approach is built upon the concept that MLLMs can understand and follow instructions provided in natural language, thereby enhancing their performance on the target task. The benefits of IT in efficient MLLMs are manifold. Firstly, it enables the model to adapt to a wide range of tasks with minimal changes to its architecture or training data. This makes it a flexible and efficient approach for fine-tuning on diverse tasks. Secondly, IT allows for better generalization, as the model learns to follow instructions and apply its knowledge to new and unseen tasks.

The IT stage is typically conducted within the paradigm of Supervised Fine-Tuning (SFT). SFT datasets are often derived from a portion of the pre-training data, which is transformed into an instruction-based format, presented in the form of single-turn or multi-turn dialogue structures. Given an image X_v and its caption, a conversation data $(X_q^1, X_a^1, \ldots, X_q^T, X_a^T)$ can be generated, where T is the total number of turns. Typically, we can organize the data into a sequence of instructions and responses following [7], where the instruction $X_{instruct}^t$ at the t-th turn as:

$$X_{instruct}^t = \begin{cases} \text{Randomly choose } [X_q^1, X_v] \textit{ or } [X_v, X_q^1], & \text{the first turn } t = 1 \\ X_q^t. & \text{the remaining turns } t > 1 \end{cases} \tag{5}$$



(a) Expert System (b) Modular Training Scheme (c) Mixture-of-Modality Adaptation

Figure 15: Comparison of different multimodal adaptation schemes for LLMs in **LaVIN** [50].

With this multimodal instruction-following sequence, IT can be performed by using the same auto-regressive training objective as that of the pre-training stage. A prevalent strategy involves maintaining the visual encoder weights in a fixed state while continuing to update the pre-trained weights of both the projector and the SLM during the IT process.

Efficient IT Current IT solutions are prohibitively expensive, requiring optimization of a large number of parameters and additional large-scale training. LaVIN [50] introduces an innovative and cost-effective solution for efficient instruction tuning of MLLMs. The Mixture-of-Modality Adaptation (MMA) in LaVIN uses lightweight modules to bridge the gap between LLMs and VL tasks. This also facilitates the joint optimization of vision and language models. The actual cost of implementing LaVIN is remarkably low, for instance, it only requires 1.4 training hours with 3.8M trainable parameters. HyperLLaVA [51] studies the under-explored dynamic tuning strategy for MLLMs and leverages the visual and language-guided dynamic tuning for the projector and LLM in two-stage training.

5.3 Diverse Training Steps

The traditional two-stage strategy, which demands the manual assignment of various adjustable parameters and dataset combinations to different training stages, can be a laborious task. To mitigate this, SPHINX-X[14] devises a single-stage, all-encompassing training pipeline that impartially treats all gathered datasets and consistently converts them into multi-modal, multi-turn dialogue formats. Throughout this unified training phase, all parameters except vision ecnoders within SPHINX-X are activated. Cobra[13] also argues that the initial phase of pre-alignment may not be requisite, with the model remaining underfitted even post-finetuning. Consequently, it discards the pre-alignment stage, opting instead to directly finetune the entire SLM backbone along with the projector. TinyGPT-V[28] training process consists of four stages: an initial pre-training stage for vision-language understanding, a second stage for refining image modality processing, a third stage for human-like learning through fine-tuning, and a fourth stage for multi-task learning to enhance its conversational abilities as a chatbot.

5.4 Parameter Efficient Transfer Learning

Several studies adopt Parameter-Efficient Fine-Tuning (PEFT) techniques for transfer learning, like LoRA [161], to safeguard against the loss of pre-trained knowledge. Efficient Attention Skipping (EAS) module[52] proposes a novel parameter and computation-efficient tuning method for MLLMs to retain the high performance and reduce both parameter and computation expenditures on downstream tasks. MemVP [53] argues that this transfer learning paradigm still exhibits inefficiency since it significantly increases the input length of the language models. Visual prompts in MemVP are concatenated with the weights of Feed Forward Networks for visual knowledge injection to reduce the training time and inference latency of the finetuned MLLMs and surpass the performance of previous PEFT methods.

6 Data and Benchmarks

In this section, we provide an overview of the data and benchmarks used for training and evaluating efficient MLLMs. We discuss the significance of pre-training data, instruction-tuning data, and the

benchmarks employed to assess the performance of these models. The discussion highlights the importance of diverse and high-quality datasets in achieving robust and accurate MLLMs, as well as the various strategies employed to generate and refine these datasets. Furthermore, we present a comprehensive comparison of MLLM performance across established benchmarks, emphasizing the need for a thorough evaluation to ensure the effectiveness of these models in real-world applications.

6.1 Pre-Training Data

Pre-training data primarily serve two critical objectives: (1) promoting the integration of various modalities and (2) conveying comprehensive knowledge. Large-scale image-caption pair datasets naturally fulfill these requirements. Firstly, they predominantly originate from the internet, providing an extensive data volume with a broad knowledge coverage. Secondly, the direct alignment between the two modalities is beneficial for training modality projectors. However, captions in such corpora are often brief and contain noise, which can be refined and filtered using automated methods, such as employing the CLIP [13] model to eliminate image-text pairs with low similarity scores. A summary of frequently used pre-training datasets can be found in Figure2.

Dataset Name	X Modality	#.X	#.T	#.X-T	Representative Publications
CC3M [162]	Image	3.3M	3.3M	3.3M	TinyGPT-V[28],MM1[30]
CC12M [163]	Image	12.4M	12.4M	12.4M	MM1[30]
SBU [164]	Image	1M	1M	1M	TinyGPT-V[28]
LAION-5B [165]	Image	5.9B	5.9B	5.9B	TinyGPT-V[28]
LAION-COCO[166]	Image	600M	600M	600M	Vary-toy [27]
COYO [167]	Image	747M	747M	747M	MM1[30]
COCO Caption[168]	Image	164K	1M	1M	Vary-toy [27]
CC595k [7]	Image	595K	595K	595K	MobileVLM [20],LLaVA-Phi [21], LLaVA-Gemma [31],Mini-Gemini [26]
RefCOCO[169]	Image	20K	142K	142K	Vary-toy [27]
DocVQA[170]	Image	12K	50K	50K	Vary-toy [27]
LLava-1.5-PT[54]	Image	558K	558K	558K	Imp-v1 [22],MoE-LLaVA [25], Vary-toy [27],Mipha [32], VL-Mamba [18],Tiny-LLaVA [23]
ShareGPT4V-PT [55]	Image	1246K	1246K	1246K	Tiny-LLava [23],MobileVLM V2 [17]
ShareGPT4 [55]	Image	100K	100K	100K	ALLaVA [29]
Bunny-pretrain-LAION-2M[24]	Image	2M	2M	2M	Bunny [24]
ALLaVA-Caption-4V [29]	Image	715K	715K	715K	Mini-Gemini [26], ALLaVA[29]
MMC4 (Interleaved) [171]	Image	571M	43B	101.2M (Instances)	DeepSeek-VL [34]
Obelics (Interleaved)[172]	Image	353M	115M	141M (Instances)	MM1[30]

Table 2: The statistics for common MLLM PT datasets.#.X represents the quantity of X, #.T represents the quantity of Text, and #.X-T represents the quantity of X-Text pairs, where X can be Image, Video, or Audio.

A growing number of studies have investigated the production of high-quality fine-grained pre-trained data by leveraging powerful MLLMs like GPT-4V. These datasets typically offer more detailed and accurate image descriptions compared to their coarse-grained counterparts, enabling a closer alignment of image and text modalities. However, this method often requires the use of commercial MLLMs, leading to increased costs and a smaller data volume. ShareGPT4V[55] addresses this issue by first training a captioner on 100K GPT-4V-generated data and then expanding the dataset to 1.2M using the pretrained captioner. Moreover,VILA's[49] findings indicate that incorporating interleaved pre-training data proves advantageous, while solely relying on image-text pairs is suboptimal in achieving the desired outcomes.

6.2 Instruction-Tuning Data

Instruction tuning(IT) is a crucial step in refining efficient MLLMs' capacity to accurately interpret user instructions and effectively carry out the desired tasks. This procedure bears a strong connection to the concept of multi-task prompting.

A summary of frequently used pre-training datasets can be found in Table.3. High-quality IT data can be derived from task-specific datasets. For instance, consider a sample from VQA datasets where the input includes an image and a natural language question, and the output is the text-based answer to the question based on the image. This could easily form the multimodal input and response of the instruction sample. The instructions, or task descriptions, can be obtained either through manual creation or semi-automatic generation with the help of GPT. In addition to utilizing publicly available task-specific datasets, SPHINX-X[14] assembles a dataset focused on OCR from a wide range of PDF data sourced from the internet. Specifically, it begins by gathering a large-scale PDF dataset from the web. It then obtains the rendering results of each page in the PDF file, while simultaneously saving all text annotations along with their respective bounding boxes. Ultimately, these elements are converted into a unified question-answering format.

While multi-task datasets provide an abundant source of data, they may not always be suitable for complex real-world situations, such as engaging in multi-turn conversations. To address this challenge, some research has explored the use of self-instruction by leveraging LLMs to generate text-based or multimodal instruction-following data from a limited number of hand-annotated samples. SPHINX-X[14] assembles a rich multi-domain dataset with fine-grained correspondence between images and texts.It gathers images from diverse sources and then employs annotations to apply various markers onto the original images. By prompting GPT-4V with these marked images and tailored domain-specific guidelines, the system generates captions that offer an image overview, regional details and object relationships insight. During the training process, SPHINX-X utilizes the unaltered images rather than the marked ones. ALLaVA[29] propose to distill a caption and a QA pair for an image within a single session. Specifically, it prompts GPT-4V with an image, and ask it to first generate a fine-grained caption then a VQA pair.

Additionally, excluding multimodal instructional data, conversations solely based on language between users and assistants can significantly contribute to enhancing a model's conversational expertise and responsiveness to directives.For example, VILA's[49] research demonstrates that integrating text-only instructional data with image-text data during the fine-tuning process not only mitigates the decline in performance for text-only tasks but also enhances the accuracy of MLLM-related tasks.

Dataset Name	Type	I→O	Source	Method	#.Instance	Representative Publications
LLaVA's IT [7]	SFT	I+T→T	MS-COCO[173]	Auto.	150K	MobileVLM [20], LLaVA-Phi [21], Mini-Gemini [26], Vary-toy [27], TinyGPT-V [28], Imp-v1 [22], ALLaVA [29],SPHINX-X [14], LLaVA-Gemma [31],MM1 [30]
ShareGPT4V's IT[55]	SFT	I+T→T	LCS, COCO [173], SAM [174], TextCaps[175], WikiArt [176]	Auto.+Manu.	–	Tiny-LLaVA [23], Mini-Gemini [26], MM1 [30], DeepSeek-VL [34], SPHINX-X [14]
LLaVA-1.5's IT [54]	SFT	I+T→T	LLaVA[7],Visual Genome[177], VQAv2[57], ShareGPT[178], A-OKVQA[179],TextCaps[175], GQA[59], OKVQA[180], OCRVQA, RefCOCO[169, 181]	Auto.+Manu.	665K	Tiny-LLava [23], VL-Mamba [18], Cobra [13], LLaVA-Gemma [31], Mipha [32], MoE-LLaVA [25]
LRV-Instruct [182]	SFT	I+T→T	Visual Genome[177]	Auto.	300K	MoE-LLaVA [25], Cobra[13]
LVIS-INSTRUCT-4V[56]	SFT	I+T→T	LVIS [183]	Auto.	220K	MoE-LLaVA [25], SPHINX-X [14], DeepSeek-VL [34], Cobra[13]
LAION GPT4V[184]	SFT	I+T→T	LAION [166]	Auto.	12.4k	Mini-Gemini [26], SPHINX-X [14], DeepSeek-VL [34]
MiniGPT-4's IT [10]	SFT	I+T→T	CC3M [162], CC12M [163]	Auto.	5K	TinyGPT-V [28],
SVIT [185]	SFT	I+T→T	MS-COCO[173], Visual Genome[177]	Auto.	3.2M	MoE-LLaVA [25]
Bunny-695K [24]	SFT	I+T→T	SVIT-mix-665K[185], WizardLM-evol-instruct-70K[186]	Auto.	695K	Bunny [24]
GQA[59]	SFT	I+T→T	Visual Genome[177]	Auto.	22M	MM1 [30], SPHINX-X [14], LLaVA-Gemma [31]

Table 3: The statistics for common MLLM IT datasets. I→O:Input to Output Modalities,T:Text.

6.3 Benchmarks

With the aim of delivering an all-encompassing performance evaluation, we have assembled a table that demonstrates the effectiveness of 22 MLLMs across 14 well-established VL benchmarks, as depicted in Table.4. Additionally, for further reference, we have incorporated a comparison of results from 13 prominent and larger MLLMs.

Model	LLM Backbone	VQA^{v2}	GQA	SQA^{I}	VQA^{T}	VizWiz	MMMU	MathV	MME^{P}	MME^{C}	MMB	SEED	POPE	LLAVA^{W}	MM-Vet
Flamingo [16]	Chinchilla-7B	-	-	-	-	28.8	-	-	-	-	-	-	-	-	-
BLIP-2 [15]	Flan-T5XXL(13B)	65.0	44.7	61.0	42.5	19.6	-	-	1293.8	290.0	-	-/46.4	85.3	38.1	22.4
LLaVA [7]	Vicuna-13B	-	41.3	-	38.9	-	-	-	-	-	-	-	-	-	-
MiniGPT-4 [10]	Vicuna-13B	-	30.8	-	19.4	-	-	-	-	-	-	-	-	-	-
InstructBLIP [8]	Vicuna-13B	-	49.5	63.1	50.7	33.4	-	-	1212.8	291.8	-	-	78.9	58.2	25.6
Qwen-VL-Chat [187]	Qwen-7B	78.2*	57.5*	68.2	61.5	38.9	35.9/32.9	-	1487.5	360.7	60.6	-/58.2	-	-	-
LLaVA-1.5 [54]	Vicuna-1.5-13B	80.0*	63.3*	**71.6**	61.3	53.6	-	-	1531.3	295.4	67.7	-/68.2	85.9	**70.7**	35.4
MiniGPT-v2-Chat [9]	LLaMA-2-Chat-7B	-	58.8	-	52.3	42.4	-	-	-	-	-	-	-	-	-
InternVL-Chat [5]	Vicuna-13B	81.2*	**66.6***	-	61.5	58.5	-	-	**1586.4**	-	-	-	**87.6**	-	-
Emu2-Chat [6]	LLaMA-33B	**84.9***	65.1*	-	66.6*	54.9	-/34.1	-	-	-	-	62.8	-	-	48.5
Gemini Pro [2]	-	71.2	-	-	74.6	-	47.9/–	45.2	-	436.79	73.6	–/70.7	-	-	64.3
Gemini Ultra [2]	-	77.8	-	-	**82.3**	-	**59.4/–**	**53.0**	-	-	-	-	-	-	-
GPT4V [1]	-	77.2	-	-	78.0	-	56.8/55.7	49.9	-	**517.14**	**75.8**	**67.3/69.1**	-	-	**67.6**
MobileVLM [20]	MobileLLaMA (2.7B)	-	59.0*	61.0	47.5	-	-	-	1288.9	-	59.6	-	84.9	-	-
LLaVA-Phi [21]	Phi-2 (2.7B)	71.4*	-	68.4	48.6	35.9	-	-	1335.1	-	59.8	-	85.0	-	28.9
Imp-v1 [22]	Phi-2 (2.7B)	79.5	-	70.0	59.4	-	-	-	1434.0	-	66.5	-	**88.0**	-	33.1
TinyLLaVA [23]	Phi-2 (2.7B)	79.9*	62.0*	69.1	59.1	-	-	-	1464.9	-	66.9	-	86.4	75.8	32.0
Bunny [24]	Phi-2 (2.7B)	79.8	62.5	70.9	-	-	38.2/33.0	-	1488.8	289.3	68.6	62.5/-	86.8	-	-
Gemini Nano-2 [2]	-	67.5	-	-	65.9	-	32.6/-	30.6	-	-	-	-	-	-	-
MobileVLM-v2 [17]	MobileLLaMA(2.7B)	-	61.1	70.0	57.5	-	-	-	1440.5	-	-	-	84.7	-	-
MoE-LLaVA [25]	Phi-2 (2.7B)	79.9*	62.6*	70.3	57.0	43.7	-	-	1431.3	-	68.0	-	85.7	-	35.9
Cobra [13]	Mamba-2.8B	75.9	58.5	-	46.0	**52.0**	-	-	-	-	-	-	**88.0**	-	-
Mini-Gemini [26]	Gemma-2B	-	-	-	56.2	-	31.7/29.1	29.4	1341.0	312.0	59.8	-	-	-	31.1
Vary-toy [27]	Qwen-1.8B	-	-	-	-	-	-	-	-	-	-	-	-	-	29.0
TinyGPT-V [28]	Phi-2 (2.7B)	-	33.6	-	-	24.8	-	-	-	-	-	-	-	-	-
SPHINX-Tiny [14]	TinyLlama-1.1B	-	-	-	57.8	-	-	26.4	1261.2	242.1	56.6	17.1/-	82.2	52.3	23.8
ALLaVA-Longer [29]	Phi-2 (2.7B)	-	50.0	-	50.3	-	33.2/-	-	1564.6†		64.6	-	-	71.7	35.5
MM1-3B [30]	MM1-3B	**82.5**	-	76.1	**72.9**	-	38.6/35.7	32.6	1469.4	303.1	70.8	63.9/69.4	87.6	**76.8**	**42.2**
LLaVA-Gemma [31]	Gemma-2b-it	71.4	58.7	-	-	-	-	-	1133.0	307.0	-	-	85.3	-	19.1
Mipha-3B [32]	Phi-2 (2.7B)	81.3*	**63.9***	70.9	56.6	47.5	-	-	1488.9	295.0	69.7	-	86.7	-	32.1
VL-Mamba [18]	Mamba-2.8B	76.6	56.2	65.4	48.9	-	-	-	1369.6	-	57.0	-	84.4	-	32.6
MiniCPM-V 2.0[33]	MiniCPM-2.4B	-	-	-	74.1	-	38.2/-	**38.7**	1808.6†		69.6	-	-	-	-
DeepSeek-VL [34]	DeepSeek-LLM-1B	-	-	-	-	-	32.2/-	31.1	-	-	64.6	-/66.7	87.6	-	36.8
KarmaVLM[71]	Qwen1.5-0.5B	-	-	53.86	45.25	-	-	-	-	-	55.8	-	-	47.5	-
moondream2[72]	Phi-1.5(1.3B)	77.7	61.7	-	49.7	-	-	-	-	-	-	-	-	-	-
Bunny-v1.1-4B[24]	Phi-3-Mini-4K	81.7	63.4	**76.3**	-	-	**40.2/38.8**	-	**1503.9**	**362.9**	**74.1**	**64.6/71.7**	87.0	-	-

Table 4: Comparison of mainstream MLLMs and efficient MLLMs on 14 VL benchmarks. VQA^{v2} [57]; VQA^{T}: TextVQA [58]; GQA [59]; SQA^{I}: ScienceQA-IMG [188]; VizWiz [189]; MMMU [190]; MathV: MathVista [191]; MME$^{P/C}$: the Perception/Cognition split of MME [60]; MMB: MMBench [61]; SEED: SEED-Bench [192]; POPE [62]; LLaVA^{W}: LLaVA-Bench (In-the-Wild) [7]; MM-Vet [193]. The two numbers reported in MMMU denote the performance on the val and test split, respectively. The two numbers reported in SEED denote the performance on the whole SEED-Bench and the image part, respectively. † denotes the combined points of two splits. * indicates that training images of the datasets are observed during training.The **red** denotes the highest result of efficient MLLMs, and the **blue** denotes that of large-scale MLLMs.

7 Applications

From the preceding analysis, it's clear that many efficient MLLM approaches evaluate their performances across a range of scenarios, like VQA, visual grounding, image segmentation, *etc*. However, it's also crucial to explore these efficient architectures in well-established tasks to achieve their ultimate performance. Therefore, we have chosen to introduce several downstream tasks, such as medical analysis, document understanding, and video comprehension.

7.1 Biomedical Analysis

Due to the high cost of annotating biomedical data, foundation models are poised to become a new paradigm in biomedicine, achieving state-of-the-art results on many applications, including medical question answering [194] and medical image classification [195]. Recently, multimodal generative AI has emerged as an exciting frontier in the biomedical domain, expanding the application scope from single-modality to multi-modality, such as VQA and radiology report generation.

The mixture of Expert Tuning has effectively enhanced the performance of general MLLMs with fewer parameters, yet its application in resource-limited medical settings has not been fully explored. MoE-TinyMed [64] is a model tailored for medical applications that significantly lower parameter demands. LLaVA-Rad [63] is a state-of-the-art tool that demonstrates rapid performance on a single V100 GPU in private settings, making it highly applicable for real-world clinical scenarios. It employs a modular approach, integrating unimodal pre-trained models and emphasizing the training of lightweight adapters. As a result, LLaVA-Rad outperforms larger models such as GPT-4V and Med-PaLM in terms of standard metrics, showcasing its superior efficiency and effectiveness.

7.2 Document Understanding

Documents or charts serve as a crucial source of information, offering an intuitive visualization of data in various forms. They have become an indispensable part of information dissemination, business decision-making, and academic research. However, current chart understanding models still face two primary limitations: (1) The considerable number of parameters makes training and deployment challenging. For instance, ChartLlama [196], a 13-billion-parameter model, is difficult to deploy on a single consumer-grade GPU. (2) These models struggle with efficiently encoding high-resolution images, as vision transformers tend to produce lengthy feature sequences.

To address the challenges of fine-grained visual perception and visual information compression for document-oriented MLLMs. TinyChart [37] outperforms several 13B MLLMs with Program-of-Thoughts (PoT) learning and Visual Token Merging strategy while excelling in faster inference speed at the same time. TextHawk [36] explores efficient fine-grained perception by designing four dedicated components to address challenges posed by document-oriented tasks. HRVDA [66] and Monkey [65] are also large multimodal models designed to address the challenges posed by high-resolution requirements in visual document understanding tasks.

7.3 Video Comprehension

Videos provide an impressively accurate representation of how humans continuously perceive the visual world. Intelligent video understanding is vital for a variety of real-world applications, including video category classification, video captioning, and video-text retrieval. Several works like videoChat [197] and Video-LLaMA [198] are LLM-based large multimodal models for end-to-end chat-centric video comprehension. However, these methods can only take in a limited number of frames for short video understanding.

To address the computational challenges associated with processing long videos due to the excessive number of visual tokens, several approaches have been developed. mPLUG-video [67] is designed for video understanding tasks and begins with a TimeSformer-based video encoder to extract features from sparsely sampled video frames effectively, followed by a visual abstractor module to reduce sequence length. Video-LLaVA [44] excels in various video understanding tasks by unifying visual representations of images and videos into a single language feature space before projection. This approach enables effective learning of multi-modal interactions with LanguageBind [93]. LLaMA-VID [69] addresses this issue by representing each frame with two distinct tokens, namely context token and content token. The context token encodes the overall image context based on user input, whereas the content token encapsulates visual cues in each frame. This dual-token strategy significantly reduces the overload of long videos while preserving critical information. Instead of trying to process more frames simultaneously like most existing work, MA-LMM [68] proposes to process videos in an online manner and store past video information in a memory bank to reference historical video content for long-term analysis without exceeding LLMs' context length constraints or GPU memory limits.

8 Discussion and Conclusion

8.1 Limitations and Future work

The development of efficient MLLMs is still in its nascent stage, and there is ample room for improvement. We summarize the current state of affairs as follows:

- At present, efficient MLLMs face challenges in processing extended-context multimodal information, and they are typically limited to accepting single images. This constrains the advancement of more sophisticated models capable of handling an increased number of multimodal tokens. Such models would be beneficial for applications like comprehending lengthy videos and analyzing extensive documents that incorporate a mix of images and text, creating more versatile and powerful systems.
- The predominant efficient MLLMs mainly support dual input modalities - images and texts, and a singular output modality - text. However, the tangible world encompasses a more extensive array of modalities. By expanding the scope of efficient MLLMs to accommodate

a richer diversity of input modalities, and augmenting their generative capacities, we can significantly bolster their multifunctionality and widen their applicability.

- There are two principal pathways to fortify efficient MLLM models. Firstly, the incorporation of a more varied set of lightweight LLMs can render the design of MLLMs more adaptable, facilitating their customization to cater to a broad spectrum of requirements. Secondly, leveraging high-quality instruction tuning datasets can empower efficient MLLMs to better comprehend and implement a vast array of instructions, thereby amplifying their zero-shot learning capabilities.
- The development of embodied agents capable of deployment on edge devices represents a crucial application prospect for efficient MLLMs. An agent possessing specialized knowledge and the capability to interact with the real world has far-reaching implications, potentially revolutionizing fields such as robotics, automation, and artificial intelligence.

8.2 Conclusion

In this study, we take a deep dive into the realm of efficient MLLM literature, providing an all-encompassing view of its central themes, including foundational theories and their extensions. Our goal is to identify and highlight areas that require further research and suggest potential avenues for future studies. We aim to provide a comprehensive perspective on the current state of efficient MLLM, with the hope of inspiring additional research. Given the dynamic nature of this field, it's possible that some recent developments may not be fully covered. To counter this, we've set up a dedicated website that uses crowdsourcing to keep up with the latest advancements. This platform is intended to serve as a continually updated source of information, promoting ongoing growth in the field. Due to space constraints, we can't cover all technical details in depth but have provided brief overviews of the key contributions in the field. In the future, we plan to continuously update and enhance the information on our website, adding new insights as they come to light.