VisCo: Leveraging Large Language Models as Intrinsic Encoders for Visual Token Compression

Yupeng Zheng
yupengzheng@mail.ustc.edu.cn
Anhui Province Key Laboratory of Digital Security,
University of Science and Technology of China
Hefei, China

Kai Zou
kzou@mail.ustc.edu.cn
Anhui Province Key Laboratory of Digital Security,
University of Science and Technology of China
Hefei, China

Bin Liu*
flowice@ustc.edu.cn
Anhui Province Key Laboratory of Digital Security,
University of Science and Technology of China
Hefei, China

Nenghai Yu
ynh@ustc.edu.cn
Anhui Province Key Laboratory of Digital Security,
University of Science and Technology of China
Hefei, China

Abstract

Vision-language models (VLMs) process large numbers of visual tokens, resulting in substantial inference latency and memory overhead. This has motivated extensive research on visual token compression. While training-free strategies rely on heuristic metrics and suffer significant performance degradation under high compression ratios, many training-based methods introduce external compression modules that force the VLM backbone to adapt, incurring substantial retraining cost and compromising VLMs' priors. Effective visual token compression hinges on strong information encoding, a capability already present in pretrained VLMs but underutilized by existing approaches. Motivated by this, we propose VisCo, a training-efficient self-compression framework that reuses the pretrained VLM itself as an intrinsic compressor. VisCo is a parameter-sharing autoencoder that compresses visual information using a small set of memory tokens and transfers hierarchical information from encoding to decoding. Experiments show that VisCo surpasses prior methods across all evaluated compression ratios, with larger gains under more aggressive compression, and remains stable even in the extreme single-token setting. Moreover, when combined with the original visual tokens, the learned memory tokens can even improve the base model, suggesting that VisCo captures complementary representations beyond compression.

CCS Concepts

• **Computing methodologies** → **Computer vision**.

Keywords

Token Compression, Vision Language Models, Autoencoder

1 Introduction

Large language models (LLMs)[4, 14, 38] have demonstrated strong capabilities in natural language understanding and generation. Recently, by jointly modeling visual information and text, the powerful priors of LLMs have been successfully transferred to visual domains, giving rise to vision-language models (VLMs)[2, 20, 29, 39] that can handle a wide range of complex tasks such as visual question answering and multimodal reasoning. However, high-resolution images are typically encoded into a large number of visual tokens, which dramatically increases the cost of self-attention computation and key–value (KV) cache storage[19], thereby severely limiting the deployment of VLMs in resource-constrained and real-time scenarios[42].

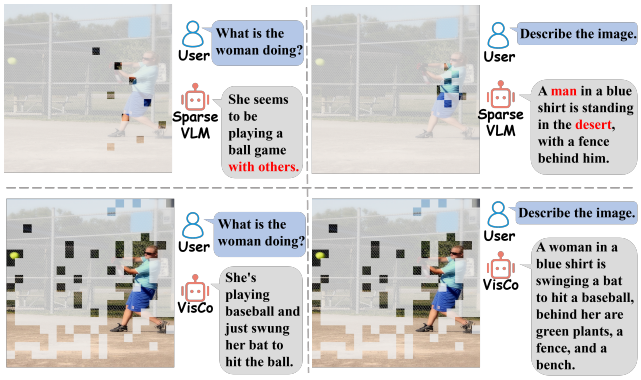

Figure 1: Comparison of SparseVLM and our method on Qwen2-VL-7B with only 9 retained visual tokens. The top two figures show the tokens preserved by SparseVLM for two different questions. The bottom figures show the top 100 tokens receiving the highest attention from the memory tokens in our method.

Prior studies have shown that visual inputs contain substantial redundancy [25, 33]. Inspired by such observations, numerous methods[3, 8, 35, 36, 40, 41, 43, 47, 48] have been proposed to compress visual tokens. Training-free methods typically perform heuristic compression on VLMs. Attention-based approaches represented by FastV[8] and SparseVLM[48] estimate token importance from text-to-vision attention in the LLM. However, as noted by [47], they are prone to attention shift, which can induce hallucinations [18]. Another line of work performs similarity-based pruning inside the vision encoder. Methods such as DART [40] and VisionZip [41] remove tokens based on local feature similarity. While effective at eliminating local redundancy, they often fail to preserve text-guided global semantics [26]. As a result, both paradigms degrade sharply under high compression ratios. As shown in Fig. 1, with only 9

*Corresponding author.

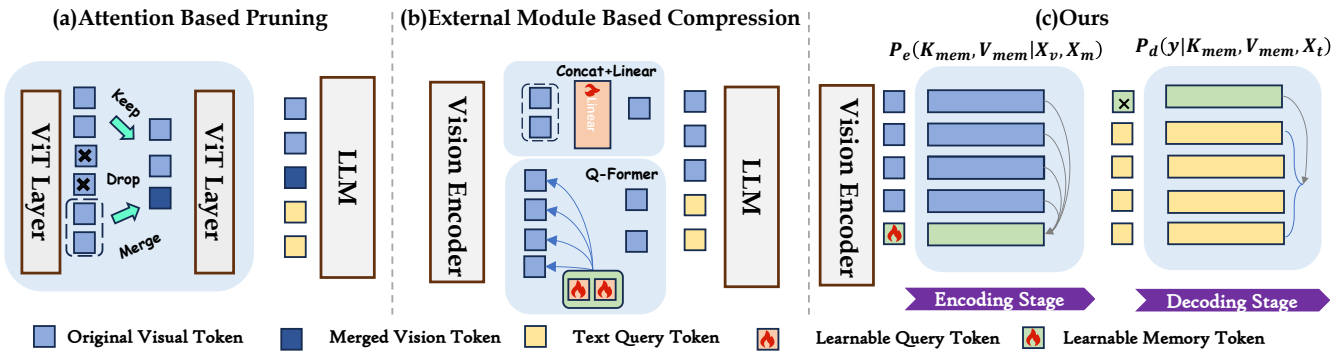

Figure 2: Comparison of Vision-Token Compression methods in the Vision Encoding Stage.

tokens, SparseVLM loses details and global context, and its token selection is highly question-dependent. In contrast, our method captures both the overall scene and key regions with very few tokens, leading to the correct answer. In contrast, training-based methods [7, 16, 21, 22, 44] improve robustness under tighter token budgets, but rely on additional compression modules or substantial retraining. Consequently, external-module-based methods often still provide limited gains under aggressive compression, while stronger methods typically come at the cost of much heavier training and reduced plug-and-play flexibility.

Visual token compression is essentially an encoding process that maps dense visual information into a compact token set while preserving essential semantics. This process hinges on strong information-encoding ability, which pretrained VLMs inherently possess. Therefore, rather than relying on heuristic pruning rules or external compression modules that force the VLM to undergo substantial retraining for adaptation, we argue that better exploiting the priors of pretrained VLMs is key to effective visual token compression. In particular, given the strong priors already embedded in pretrained VLMs, an effective compression method should follow the native processing pattern of the VLM itself and achieve compression with only lightweight adaptation.

Motivated by this, we propose **VisCo**, an autoencoder that leverages priors of VLMs to achieve **Vi**sual token **s**elf-**Co**mpression during the vision encoding stage. Unlike prior approaches that either rely on attention and similarity for compression (Fig. 2(a))[35, 41] or employ external modules with high retraining cost (Fig. 2(b))[7, 21], VisCo reuses the VLM itself (Fig. 2(c)) as the compressor: a small set of learnable memory tokens is introduced to interact with visual tokens in the encoder, while hierarchical visual information is passed from the encoder to the decoder to preserve task-relevant semantics under a reduced token budget. This design keeps the backbone intact and requires only lightweight adaptation. Our main contributions are summarized as follows:

(1). An asymmetric intrinsic self-compression framework. We propose VisCo, which uses the VLM itself to compress its own visual tokens with lightweight training while keeping the pretrained backbone unchanged.

(2). A hierarchical information-passing mechanism. We introduce a hierarchical mechanism to pass multi-granularity semantics from encoder layers to the decoder under compression.

(3). Comprehensive evaluation and strong performance under extreme compression. Extensive experiments on 3 VLM backbones and 6 benchmarks show that VisCo consistently outperforms existing methods, especially at high compression ratios. We further show that the learned compressed representations capture complementary information beyond compression alone.

2 Related Work

2.1 Vision Language Models

Recent advances in LLMs[5, 31, 38] have driven the rapid development of VLMs, which typically consist of a vision encoder, a modality alignment module, and an LLM backbone. Representative VLMs, such as LLaVA[29], BLIP-2[21], and InstructBLIP[11], typically integrate pretrained visual encoders[32, 46] with LLM backbones via lightweight modality alignment modules, and many of these modular frameworks are trained under a two-stage paradigm that first establishes cross-modal alignment and subsequently equips the model with multimodal instruction-following capabilities. Despite their strong multimodal reasoning ability, current VLMs still struggle with fine-grained perception such as OCR and high-resolution detail understanding[37]. To alleviate these limitations, recent efforts[28] have resorted to higher input resolutions to enhance visual perception and reduce hallucinations, consequently introducing substantially more visual tokens and imposing heavier computational overhead. To reduce the computational overhead, several recent VLMs[9, 10, 39, 42] have incorporated built-in visual compression mechanisms, including the dynamic-resolution design of Qwen2-VL[39] and the token reduction strategy adopted in InternVL[10]. Yet the compression achieved by these architectures is still limited, which suggests that more dedicated visual token compression methods remain necessary for further improving efficiency without sacrificing fine-grained semantics or downstream reasoning performance.

2.2 Token Compression Methods

Training-free methods typically prune or merge visual tokens based on cross-modal attention scores or token similarity. Early approaches such as FastV[8] and SparseVLM[48] mainly perform attention-based token pruning, but as the token budget becomes tighter, they often suffer substantial performance degradation due to the loss of

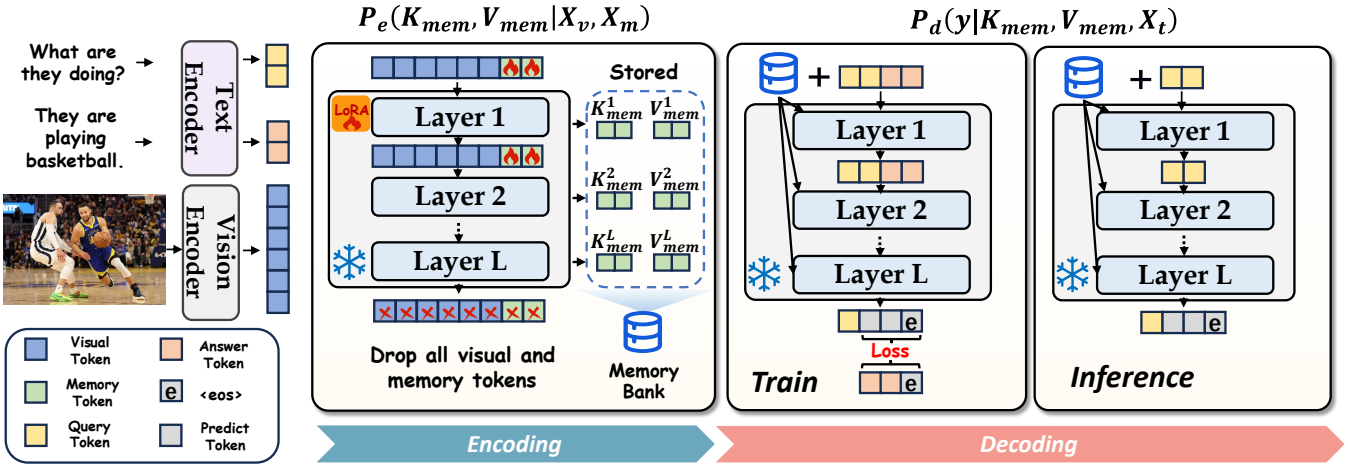

Figure 3: Illustration of the VisCo framework. During the encoding phase, hierarchical KV values corresponding to memory tokens are stored in a memory bank; during the decoding phase, these KV values are retrieved and hierarchically populated into the KV cache.

fine-grained details and global semantic information. To alleviate this issue, methods such as VisionZip[41] introduce token merging to preserve more global semantics during compression, yet their gains remain limited under high compression ratios.

Overall, training-free methods often remain competitive under mild compression, but such settings are not particularly challenging. Prior work[26] has shown that, at low compression ratios, even simple input downsampling can outperform existing methods. Thus, the central challenge in visual token compression is to preserve semantic fidelity under aggressive compression.

To address this challenge, recent training-based methods can be roughly divided into two categories. One line of work, represented by PruMerge[35] and ACCM[13], introduces lightweight compression modules with fine-tuning-level training cost. While efficient, such methods still provide only limited gains under aggressive compression. Another line of work adopts heavier training pipelines to improve performance under extreme token budgets. Earlier methods such as QueCC[22] and Matryoshka Query[16] rely on external compression modules or architectural modifications, leading to higher training cost while still requiring the VLM to adapt to a new compression mechanism. More recent concurrent works, such as VoCo-LLaMA[44] and C&C[6], begin to explore compression within the VLM itself. However, they still rely on costly alignment, instruction tuning, or more specialized compression training, and therefore achieve strong performance at the cost of substantially modifying the original operating regime of the pretrained VLM. Different from both lines of work, VisCo pursues lightweight adaptation while directly leveraging the pretrained VLM itself for fine-grained hierarchical visual token self-compression. By following, rather than rewriting, the pretrained VLM's native processing pattern, VisCo achieves performance competitive with heavyweight methods while retaining the flexibility and efficiency of lightweight adaptation.

3 Method

We propose VisCo, a framework that compresses visual tokens by exploiting the priors of the VLM itself. As illustrated in Fig. 3, VisCo is an asymmetric VLM autoencoder that introduces a lightweight trainable module in the encoding stage while keeping the entire VLM frozen during decoding.

3.1 Overall Architecture: Asymmetric VLM Autoencoder with Shared Parameters

VisCo is designed as a general solution for advanced VLMs such as Qwen2-VL[39] and LLaVA-1.5-7B[27]. Let the pretrained VLM backbone be denoted by $\Phi(\cdot)$. In the encoding stage, we augment the VLM with LoRA[15] adapter θ_{LoRA} and introduce a small set of learnable memory tokens X_m. The encoder can then be written as $\Phi_E(\cdot; X_m, \theta_{\text{LoRA}})$, where the trainable parameters only include X_m and θ_{LoRA}. In the decoding stage, we remove the LoRA adapters used in the encoder and directly reuse the shared pretrained LLM backbone as the decoder. This asymmetric design adapts only the encoder while keeping the decoder frozen, so that VisCo requires only lightweight adaptation of the pretrained VLM's existing capabilities to the compression setting rather than relearning the whole multimodal generation process from scratch, which is different from methods like VoCo-LLaMA[44].

3.2 LLM-based Encoding Process

3.2.1 Sequence Construction under Causal Masking. Following [29], given an image I^{origin}, we first utilize the function f_{resize} that rescales the image to a target resolution to obtain I. The vision encoder f_v and the multimodal projector g encode the origin I into a sequence of visual tokens $X_v = g(f_v(I)) \in \mathbb{R}^{N_v \times D}$ where N_v is the number of visual tokens and D is the feature dimension. We then append N_m learnable memory tokens $X_m \in \mathbb{R}^{N_m \times D}$, where $N_m \ll N_v$, and form the joint input sequence $X = [X_v; X_m] \in \mathbb{R}^{(N_v+N_m)\times D}$. VisCo strictly follows the standard causal masking mechanism used in

Table 1: Comparison on LLaVA-1.5-7B under different token budgets. Avg. denotes the average percentage of performance maintained relative to the original LLaVA-1.5-7B. Methods marked with † are stronger references that are not directly comparable due to stronger training or different experimental settings.

Method	Token	GQA	MMB	MMB-CN	MME	PoPE	MMVet	Avg.
LLaVA-1.5-7B	576	62.0	64.3	58.3	1510.7	85.9	31.1	100.0%
FastV (ECCV24)	32	41.5	37.8	33.2	884.6	32.5	20.7	57.6%
SparseVLM (ICML25)	32	48.3	51.4	40.6	1046.7	67.9	18.6	72.6%
PruMerge+ (ICCV25)	32	51.1	56.8	47.0	940.8	70.9	21.4	77.5%
DivPrune (CVPR25)	32	54.9	57.6	49.1	**1284.9**	81.5	26.3	87.2%
VisPruner (ICCV25)	32	52.2	58.4	52.7	1271.0	72.7	**28.8**	87.8%
VisCo	32	**58.5**	**62.3**	**57.2**	1152.9	**81.9**	27.9	**91.8%**
VoCo-LLaMA (CVPR25)†	32	60.2	59.4	-	-	-	-	-
PruMerge+ (ICCV25)	1	26.4	13.7	13.7	568.9	40.4	12.5	35.4%
VisPruner (ICCV25)	1	41.8	22.4	25.8	764.4	49.0	12.0	48.8%
VisCo	1	**58.2**	**59.0**	**52.4**	**1191.2**	**78.2**	**20.7**	**85.3%**
Matryoshka (NIPS24)†	1	50.8	54.4	-	1144.0	74.5	-	-
VoCo-LLaMA (CVPR25)†	1	57.0	58.8	-	1323.3	81.4	-	-

decoder-only VLMs, where a token at position i is only allowed to attend to tokens at positions $j \le i$. Therefore, by placing X_m after the visual tokens X_v in the sequence, each memory token can naturally attend to all visual tokens under the native attention pattern of the pretrained backbone. In this way, VisCo does not introduce any customized interaction rule, but instead simply leverages the model's intrinsic attention prior to let the memory tokens aggregate rich visual information from the visual tokens.

3.2.2 Hierarchical Information Aggregation. Transformer-based LLMs are known to process information in a hierarchical manner. Prior works have shown that Transformer language models exhibit layer-wise functional specialization: lower layers tend to encode surface and local syntactic patterns, middle layers focus on syntactic structure, while higher layers capture semantics and discourse-level phenomena [34]. A compact representation for visual compression should preserve not only the final semantic summary, but also the hierarchical structure of visual information. Relying only on the final encoder output would collapse this hierarchy and make it harder for the decoder to recover fine-grained details under aggressive compression. Instead of passing the output of the encoder to the decoder, which destroys the layer-wise structure of compression information, VisCo is motivated by the structural similarity between the encoder and decoder. It adopts a hierarchical compression scheme that transfers hierarchical information from the encoder to the decoder, leveraging the self-attention mechanism of LLMs. For each layer $l = 1, \ldots, L$, let $K^{(l)}$ and $V^{(l)}$ denote the key and value matrices of the self-attention computed over the joint sequence $X = [X_v; X_m]$. We construct a layer-wise memory bank by retaining only the key and value entries corresponding to the memory tokens:

$$\begin{aligned} K_{\text{mem}}^{(l)} &= K^{(l)}[N_v + 1 : N_v + N_m], \\ V_{\text{mem}}^{(l)} &= V^{(l)}[N_v + 1 : N_v + N_m]. \end{aligned} \tag{1}$$

We define $K_{\text{mem}} = \{K_{\text{mem}}^{(l)}\}_{l=1}^{L}$ and $V_{\text{mem}} = \{V_{\text{mem}}^{(l)}\}_{l=1}^{L}$, and model the encoding stage as:

$$P_e(K_{\text{mem}}, V_{\text{mem}} \mid X_v, X_m; \Phi_{\text{E}}). \tag{2}$$

Under causal masking, each memory token can attend to all preceding visual tokens. Consequently, shallow layers of $\{K_{\text{mem}}^{(l)}, V_{\text{mem}}^{(l)}\}$ capture low-level cues such as textures and local patterns, whereas deeper layers encode progressively more abstract semantic information. In practice, we collect the KV pairs associated with memory tokens at every layer into the memory bank, discard the final encoder hidden states, and rely solely on these layer-wise memory KV caches during decoding.

3.3 Hierarchical Prefix Decoding

Our encoder and decoder share the same backbone parameters. Specifically, the decoding stage reuses the pretrained VLM Φ with all LoRA adapters removed. To enable the decoder to exploit the hierarchical memory information collected during encoding, we directly populate the decoder's KV cache with the layer-wise memory bank $\{K_{\text{mem}}^{(l)}, V_{\text{mem}}^{(l)}\}_{l=1}^{L}$ and then perform autoregressive generation on top of this cache. Accordingly, we model the decoding stage as

$$P_d(y \mid K_{\text{mem}}, V_{\text{mem}}, X_t; \Phi), \tag{3}$$

where y denotes the answer sequence and $X_t \in \mathbb{R}^{N_t \times D}$ denotes the embedding sequence of textual prompt with length N_t.

Benefiting from the strong priors of the underlying VLM, we do not introduce any additional pre-training objectives. Instead, we fine-tune VisCo end-to-end with teacher forcing, which keeps the training cost low. We maximize the conditional likelihood of the ground-truth answer given the compressed memory and textual inputs. The resulting loss is

$$\mathcal{L}_{\text{FT}} = -\sum_{i=1}^{N_a} \log p_\theta\big(a_i \mid K_{\text{mem}}, V_{\text{mem}}, X_t, A_{<i}\big), \tag{4}$$

Table 2: Performance comparison of different token compression methods on Qwen2-VL-2B and Qwen2-VL-7B. Here, Avg represents the average percentage of performance maintained. ‡ indicates additional fine-tuning.

Base Model	Method	GQA	MMB	MMB-CN	MME	POPE	MMVet	Avg.
Qwen2-VL-2B	Upper Bound, 144 Tokens (100%)							
	Origin	59.8	67.3	61.8	1465.8	81.5	42.5	100.0%
	Retain 36 Tokens (↓ 66.7%)							
	FastV (ECCV24)	49.5	62.2	59.3	1289.3	68.9	26.2	84.2%
	PruMerge+ (ICCV25)	52.1	49.3	46.8	**1387.2**	70.2	28.4	80.6%
	VisionZip‡ (CVPR25)	53.2	**64.3**	56.7	1372.4	75.3	35.5	91.0%
	VisCo	**61.1**	64.0	**58.9**	1368.3	**82.9**	**35.6**	**95.2%**
	Retain 18 Tokens (↓ 88.9%)							
	FastV (ECCV24)	46.2	56.0	47.9	1159.2	49.0	18.1	70.0%
	PruMerge+ (ICCV25)	46.6	47.9	26.5	1190.7	61.0	18.1	65.1%
	VisionZip‡ (CVPR25)	48.5	55.2	41.6	1300.9	62.7	25.8	76.1%
	VisCo	**59.7**	**62.9**	**57.8**	**1358.0**	**82.4**	**28.9**	**91.4%**
Qwen2-VL-7B	Upper Bound, 144 Tokens (100%)							
	Origin	64.8	76.1	71.6	1664.8	82.6	56.1	100.0%
	Retain 36 Tokens (↓ 66.7%)							
	FastV (ECCV24)	55.4	70.7	64.6	1492.0	65.8	36.6	84.6%
	PruMerge+ (ICCV25)	59.2	71.8	63.1	1518.7	73.9	37.5	88.9%
	VisionZip‡ (CVPR25)	58.3	72.4	63.4	1530.0	75.3	39.1	89.8%
	VisCo	**62.6**	**74.0**	**70.0**	**1551.0**	**83.3**	**40.4**	94.6%
	Retain 18 Tokens (↓ 88.9%)							
	FastV (ECCV24)	52.8	65.4	59.4	1395.2	60.7	33.4	77.9%
	PruMerge+ (ICCV25)	52.8	61.2	47.7	1351.7	63.7	30.6	73.6%
	VisionZip‡ (CVPR25)	55.3	66.4	58.3	1443.8	71.7	33.9	81.3%
	VisCo	**61.8**	**70.0**	**68.8**	**1496.5**	**81.7**	**39.2**	**90.4%**

where $A = \{a_1, \ldots, a_{N_a}\}$ denotes the answer token sequence with length N_a and $A_{<i}$ represents answer tokens located before the current predicted token a_i. θ represents trainable parameters, consisting of X_m and θ_{LoRA}. Unlike Prefix-Tuning[23], which uses L per-layer MLPs to project prefix tokens into key/value prefixes, we directly reuse the encoder-side memory KV caches as hierarchical prefixes. Since the encoder-side KV caches already carry layer-wise, fine-grained information and the encoder and decoder share the same representational and attention priors via parameter sharing, the decoder can directly reuse them without any projection.

4 Experiments

In this section, we comprehensively evaluate the performance and efficiency of our solution using 6 benchmarks across 3 VLM models.

4.1 Implementation Details

We fine-tune VisCo for one epoch on a ~10% subset of LLaVA-665K [27], consisting of the first turn of each conversation. We apply LoRA to the Q/V projections with $rank = 64$ and $\alpha = 128$, using a learning rate of 5×10^{-5}. We evaluate VisCo on LLaVA-1.5-7B and Qwen2-VL (2B/7B). Notably, Qwen2-VL incorporates native 4× compression, serving as a rigorous testbed to validate VisCo's robustness on already-compact visual representations.

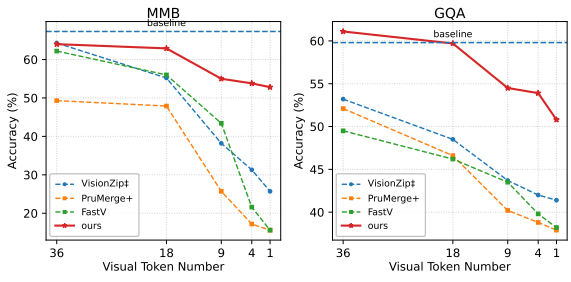

Figure 4: Ablation Study of Token Compression Rates on MMB and GQA.

4.2 Datasets

We evaluated VisCo across six multimodal benchmarks, including MME[12], MMB[30], MMB-CN, MMVet[45], GQA[17], and POPE [24]. We report standard metrics for each benchmark, including the perception score on MME, accuracy on MMB/MMB-CN

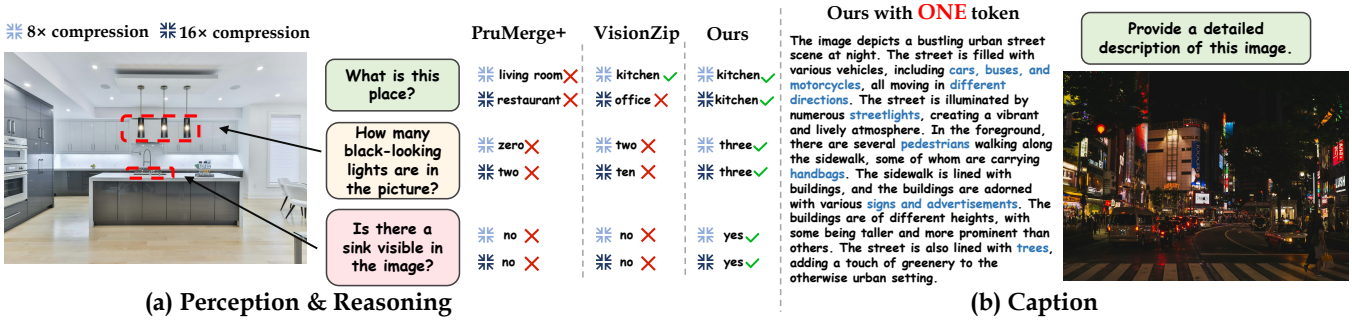

Figure 5: Qualitative evaluation under high compression ratio. (a) Perception and reasoning comparison of existing compression methods versus VisCo. (b) Caption generation from a single VisCo token. All experiments were conducted on Qwen2-VL-2B

and GQA, the GPT-4[1] assisted score on MMVet, and F1 on POPE. We also report Avg, defined as the unweighted mean of the retained-score ratio (compressed/original) across benchmarks.

4.3 Baselines

We compare VisCo against six approaches. FastV[8], SparseVLM[48], DivPrune[3], and VisPruner[47] are plug-and-play methods that can be applied directly without extra fine-tuning. Also, we include two fine-tuned baselines: PruMerge+[35] fine-tuned for 1 epoch on the full LLaVA-665K, and VisionZip‡[41] fine-tuned for 1 epoch on the same 10% subset of LLaVA-665K as VisCo. ‡ denotes VisionZip with additional fine-tuning. In addition, we further include two stronger references, VoCo-LLaMA[44] and MatryoshkaQuery[16]. Among all VLM token compression methods, VoCo-LLaMA represents a SOTA reference. However, both methods require substantially retraining of the VLM, including re-alignment and large-scale instruction tuning, and therefore are not directly comparable to our lightweight fine-tuning setting. We report both methods for reference only. Due to differences in publicly available implementations and backbone compatibility across methods, the set of comparable baselines varies slightly across VLM backbones.

Table 3: Ablation Study Comparing Value-Passing and Hierarchical KV-Pass for Encoder–Decoder Information Transfer. Score is the retained performance percentage.

Method	GQA		MMB		POPE	
	Acc.	Score	Acc.	Score	F1.	Score
Upper Bound, All 144 Tokens (**100%**)						
Origin	59.8	100%	67.3	100%	81.5	100%
Retain 36 Tokens (↓ 66.7%)						
Our-ValuePass	51.8	86.6%	59.5	88.7%	79.6	97.7%
Our-HierKVPass	61.1	102.2%	64.0	95.1%	82.9	101.7%
Retain 18 Tokens (↓ 88.9%)						
Our-ValuePass	49.6	82.9%	54.4	81.1%	77.6	95.2%
Our-HierKVPass	59.7	99.8%	62.9	93.5%	82.4	101.1%
Retain 2 Tokens (↓ 98.6%)						
Our-ValuePass	45.8	76.6%	50.1	74.7%	74.9	91.9%
Our-HierKVPass	50.8	84.9%	53.3	79.2%	79.7	97.8%

4.4 Main Results

Results on LLaVA-1.5. As shown in Table 1, we first evaluate VisCo on the classic LLaVA-1.5-7B under high-compression settings. When compressing the original 576 visual tokens to 32, VisCo still preserves 91.8% of the original performance, outperforming all directly comparable baselines and surpassing VisPruner by 4.0 points in Avg. It also achieves the best results on four benchmarks, indicating that VisCo can retain task-relevant visual information more effectively under aggressive compression. When the token budget is further reduced to a single token, the advantage of VisCo becomes even more pronounced. Despite this extreme compression ratio, VisCo still retains 85.3% of the original performance, exceeding PruMerge+ and VisPruner by 49.9 and 36.5 points in Avg., respectively. This suggests that even a very small number of memory tokens can still form compact yet effective visual representations.

For further reference, we also include two stronger training-based methods, VoCo-LLaMA and MatryoshkaQuery. Although they require substantially more large-scale retraining and are therefore not comparable to our lightweight fine-tuning setting, VisCo still consistently outperforms MatryoshkaQuery and even surpasses VoCo-LLaMA on GQA and MMB under the 1-token setting, further demonstrating its strong competitiveness in high-compression scenarios.

Results on Qwen-2-VL. To assess generality, we further benchmark VisCo using Qwen2-VL. As shown in Table 2, on Qwen2-VL-2B, VisCo retains 95.2% of the full-token performance when compressing to 36 tokens, exceeding VisionZip‡ by 4.2%. When the budget is tightened to 18 tokens, prior methods degrade substantially: even the strongest baseline on Qwen2-VL, VisionZip‡, retains only 76.1%, whereas VisCo still maintains 91.4%. On Qwen2-VL-7B, VisCo remains consistently competitive under an 8× compression ratio and outperforms VisionZip‡ by 9.1%.

Analysis. VisCo consistently outperforms directly comparable baselines and remains competitive with stronger references. Its advantage is especially evident on Qwen2-VL, which already incorporates a built-in visual compression mechanism. When handling denser visual information, existing compression methods suffer severe performance degradation, whereas VisCo remains remarkably stable. These results further validate the rationality of exploiting the intrinsic capabilities of LLMs for compression. Moreover, the

Table 4: Comparison of VisCo+, the original model, and Direct SFT on 6 benchmarks with Qwen2-VL-2B. Direct SFT denotes directly fine-tuning the original model using the same training configuration as VisCo+. The best results are in bold and the second-best results are underlined.

Method	MMB	MMB-CN	GQA	MME	PoPe	MMVet
Origin	<u>67.3</u>	61.8	<u>59.8</u>	<u>1465.8</u>	81.5	**42.5**
Direct SFT	66.4	<u>62.2</u>	57.1	1354.5	<u>82.6</u>	<u>40.3</u>
VisCo+	**69.6**	**67.8**	**59.9**	**1491.2**	**84.6**	37.2

gains are unlikely to come from fine-tuning or answer memorization. PruMerge+ and VisionZip‡ are trained on LLaVA-665K using the full set or a 10% subset, yet they still lag behind VisCo. This suggests that VisCo benefits primarily from its intrinsic memory-based compression rather than supervision. Notably, despite being trained without Chinese data, VisCo still achieves the best performance on MMB-CN, further supporting the effectiveness of our compression mechanism.

4.5 Ablation Study

4.5.1 Ablation Study of Reduction Ratios. To explore VisCo's compression limits, we compare different compression strategies on Qwen2-VL-2B across a range of compression ratios on GQA and MMB. As shown in Fig. 4, the performance of existing methods drops sharply as the number of retained visual tokens decreases, and all of them converge to a similar low accuracy when only a single token is preserved. In contrast, our method still achieves accuracies of 50.8 and 52.8 on GQA and MMB, respectively, outperforming all competing methods even when they keep 9 tokens. Additional results are provided in the supplementary material.

As shown in Fig. 5 (a), we compare VisCo with existing approaches under both 8× and 16× compression ratios. At high compression ratios, conventional methods miss fine-grained details, causing failures on complex reasoning and even hallucinations on simple scene classification due to weakened global semantics. In contrast, VisCo consistently answers all perception and reasoning tasks correctly. Fig. 5 (b) further showcases the captioning capability of VisCo. Remarkably, with only a single token, VisCo is able to generate a caption containing both detailed object-level descriptions and coherent global semantics. More examples are provided in the supplementary material.

4.5.2 Ablation Study of Hierarchical Aggregation and Decoding. To validate the effectiveness of our hierarchical KV-cache propagation scheme for compression, we implemented an alternative method: a value-passing variant that directly feeds the outputs of the encoder into the decoder. We conducted experiments on Qwen2-VL-2B under three different compression ratios using GQA, MMB and POPE. As shown in Table 3, the results demonstrate that Our-HierKVPass consistently outperforms Our-ValuePass, confirming that hierarchical information propagation indeed substantially improves model performance. Furthermore, we observe that Our-ValuePass still surpasses the other baseline methods listed in Table 2, especially at high compression rates. This suggests that the performance gains do not arise solely from hierarchical aggregation, but also from the autoencoder architecture itself, which is able to better exploit the strong priors of the LLM.

Figure 6: Visualization of attention in the encoding stage of Qwen2-VL-2B with 36 memory tokens. The transparent regions on the left highlight the top-175 visual tokens attended by the first four memory tokens, while those on the right correspond to the last four memory tokens.

4.5.3 Ablation Study of Memory Tokens as Complementary Representations. As shown in Table 2, we observe an interesting phenomenon in the main experiments. On benchmarks such as GQA and POPE, models using compressed tokens can even outperform the original model. This suggests that memory tokens are not merely a subset or summary of the original visual tokens. Instead, they provide complementary representations and encode visual information from a new perspective.

To verify this hypothesis, we propose VisCo+. It is built upon the VisCo model trained with 36 retained tokens in Table 2. During decoding, it jointly leverages the key-value pairs of the original visual tokens and the memory tokens. As shown in Table 4, we compare VisCo+, the original model, and a directly fine-tuned variant on Qwen2-VL-2B. The results show that VisCo+ consistently improves over the original model. The comparison with direct SFT further shows that the gain does not come from additional fine-tuning. These results support our claim that memory tokens are not simple compressed representations. They instead act as complementary representations that enrich the original visual features.

Table 5: Ablation study of dropping different ranges of memory tokens on Qwen2-VL-2B. The best results are in bold and the second-best results are underlined.

Method	MMB	MMB-CN	GQA	MME	PoPe
36token	**64.0**	**58.9**	**60.1**	**1395.4**	**82.9**
drop(0,17)	55.7	52.9	56.8	1305.6	76.2
drop(18,35)	<u>60.0</u>	<u>56.2</u>	<u>57.0</u>	<u>1375.6</u>	<u>81.6</u>

Table 6: Efficiency comparison under different token budgets on LLaVA-1.5-7B.

Token	Method	Compression Time (ms)	Decode Time (ms/token)	Inference Time (s)	KV Cache (MB)
576	Origin	–	21.2	5.5	288.0
128	FastV	–	20.3	5.2	78.8
	VisionZip‡	–	19.6	5.0	72.0
	VisCo	68.9	19.6	5.0	72.0
32	FastV	–	19.4	5.0	26.4
	VisionZip‡	–	18.6	4.8	18.0
	VisCo	63.7	18.6	4.8	18.0

4.5.4 How Do Memory Tokens Work? To investigate how memory tokens transfer information between the encoder and decoder, and to assess the importance of memory tokens at different positions, we first conduct an ablation study by dropping different ranges of memory tokens. As shown in Table 5, the standard VisCo uses 36 memory tokens during inference. Dropping the first 18 tokens causes a substantial performance decline, while dropping the last 18 tokens has a much smaller effect. This trend is consistent across five benchmarks.

This observation raises a natural question: are the later memory tokens redundant? To answer this, we further visualize the attention from memory tokens at different positions to the original visual tokens. As shown in Fig. 6, the first four memory tokens attend more densely to the main objects in the image, whereas the last four tokens exhibit much sparser attention and tend to focus on background regions and local details. This finding also explains the results in Table 5. Early memory tokens are more likely to encode the main content of the image, and most benchmark questions are primarily centered on the main objects. Therefore, these tokens are more critical for performance. However, this does not mean that the later tokens are redundant. Instead, they capture scattered details and background information, which helps enrich the model's overall scene perception.

4.6 Efficiency Analysis

To evaluate the efficiency of VisCo, we conduct analyses on both long-response and short-response scenarios using MMVet[45] and MME[12], respectively, on a single 48GB NVIDIA RTX 6000 GPU.

On MMVet, we fix the output length of LLaVA-1.5-7B to 256 tokens and compare VisCo with FastV and VisionZip‡. As shown in Table 6, although VisCo introduces extra compression overhead due to its additional LLM-based encoding step, it achieves the same decoding speed as VisionZip and consistently outperforms Origin and FastV under both the 128-token and 32-token settings. In long-response scenarios, the decoding acceleration offsets the compression overhead, making VisCo 0.2 s faster than FastV in both settings. VisCo also significantly reduces KV-cache storage.

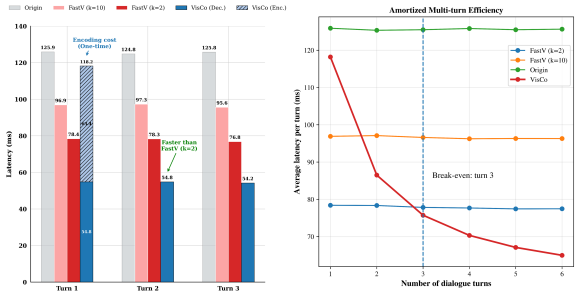

Figure 7: Multi-turn efficiency on MME with LLaVA-1.5-7B. Left: per-turn latency. Right: amortized average latency per turn, with VisCo reaching break-even at turn 3.

On MME, we evaluate efficiency in a short-answer setting, where the compression overhead becomes more noticeable. However, in practical applications, users often ask multiple questions about the same image. For methods such as FastV and SparseVLM, which rely on the current question text to compress visual tokens, the visual tokens must be compressed again for each query. In contrast, VisCo compresses the image once and reuses the cached representations across dialogue turns. As shown in Fig. 7, although VisCo already achieves lower latency than Origin in the first round of question answering, its response time is still noticeably slower than that of FastV due to the one-time compression overhead. As the number of dialogue turns increases, however, the advantage of cache reuse in VisCo gradually becomes evident, and its response speed in later turns becomes substantially faster than that of FastV. The right panel of Fig. 7 further shows that VisCo surpasses FastV in terms of average response time after only three dialogue turns.

Overall, VisCo consistently reduces KV cache memory across all evaluated scenarios, and demonstrates clear efficiency advantages in long-response and single-image multi-question scenarios.

5 Conclusion

In this paper, we present VisCo, a training-efficient visual token compression framework that leverages the pretrained VLM itself as an intrinsic compressor. By performing compression through lightweight adaptation while keeping the backbone intact, VisCo avoids the severe degradation of training-free methods under aggressive compression and the heavy retraining cost of external-module-based approaches. Extensive experiments on 3 VLM backbones and 6 benchmarks show that VisCo consistently surpasses existing methods across diverse compression ratios, while remaining effective even in the extreme one-token setting. Overall, VisCo demonstrates that leveraging the intrinsic priors of pretrained VLMs enables robust visual token compression with only lightweight training under tight token budgets. Future work will explore adaptive token allocation for different visual inputs and extend VisCo to more challenging video compression settings.

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774* (2023).

[2] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, et al. 2022. Flamingo: a visual language model for few-shot learning. *Advances in neural information processing systems* 35 (2022), 23716–23736.

[3] Saeed Ranjbar Alvar, Gursimran Singh, Mohammad Akbari, and Yong Zhang. 2025. Divprune: Diversity-based visual token pruning for large multimodal models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*. 9392–9401.

[4] Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609* (2023).

[5] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems* 33 (2020), 1877–1901.

[6] Adrian Bulat, Yassine Ouali, and Georgios Tzimiropoulos. 2025. Compress & Cache: Vision token compression for efficient generation and retrieval. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

[7] Jun Chen, Deyao Zhu, Xiaoqian Shen, Xiang Li, Zechun Liu, Pengchuan Zhang, Raghuraman Krishnamoorthi, Vikas Chandra, Yunyang Xiong, and Mohamed Elhoseiny. 2023. Minigpt-v2: large language model as a unified interface for vision-language multi-task learning. *arXiv preprint arXiv:2310.09478* (2023).

[8] Liang Chen, Haozhe Zhao, Tianyu Liu, Shuai Bai, Junyang Lin, Chang Zhou, and Baobao Chang. 2024. An image is worth 1/2 tokens after layer 2: Plug-and-play inference acceleration for large vision-language models. In *European Conference on Computer Vision*. Springer, 19–35.

[9] Zhe Chen, Weiyun Wang, Yue Cao, Yangzhou Liu, Zhangwei Gao, Erfei Cui, Jinguo Zhu, Shenglong Ye, Hao Tian, Zhaoyang Liu, et al. 2024. Expanding performance boundaries of open-source multimodal models with model, data, and test-time scaling. *arXiv preprint arXiv:2412.05271* (2024).

[10] Zhe Chen, Jiannan Wu, Wenhai Wang, Weijie Su, Guo Chen, Sen Xing, Muyan Zhong, Qinglong Zhang, Xizhou Zhu, Lewei Lu, et al. 2024. Internvl: Scaling up vision foundation models and aligning for generic visual-linguistic tasks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 24185–24198.

[11] Wenliang Dai, Junnan Li, Dongxu Li, Anthony Tiong, Junqi Zhao, Weisheng Wang, Boyang Li, Pascale N Fung, and Steven Hoi. 2023. Instructblip: Towards general-purpose vision-language models with instruction tuning. *Advances in neural information processing systems* 36 (2023), 49250–49267.

[12] Chaoyou Fu, Peixian Chen, Yunhang Shen, Yulei Qin, Mengdan Zhang, Xu Lin, Zhenyu Qiu, Wei Lin, Jinrui Yang, Xiawu Zheng, Ke Li, Xing Sun, and Rongrong Ji. 2023. MME: A Comprehensive Evaluation Benchmark for Multimodal Large Language Models. *ArXiv* abs/2306.13394 (2023). https://api.semanticscholar.org/CorpusID:259243928

[13] Mingyu Fu, Wei Suo, Ji Ma, Lin Yuanbo Wu, Peng Wang, and Yanning Zhang. 2025. Mitigating information loss under high pruning rates for efficient large vision language models. In *Proceedings of the 33rd ACM International Conference on Multimedia*. 4156–4165.

[14] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783* (2024).

[15] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. 2022. Lora: Low-rank adaptation of large language models. *ICLR* 1, 2 (2022), 3.

[16] Wenbo Hu, Zi-Yi Dou, Liunian H Li, Amita Kamath, Nanyun Peng, and Kai-Wei Chang. 2024. Matryoshka query transformer for large vision-language models. *Advances in Neural Information Processing Systems* 37 (2024), 50168–50188.

[17] Drew A Hudson and Christopher D Manning. 2019. Gqa: A new dataset for real-world visual reasoning and compositional question answering. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 6700–6709.

[18] Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of hallucination in natural language generation. *ACM computing surveys* 55, 12 (2023), 1–38.

[19] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*. 611–626.

[20] Bo Li, Yuanhan Zhang, Dong Guo, Renrui Zhang, Feng Li, Hao Zhang, Kaichen Zhang, Peiyuan Zhang, Yanwei Li, Ziwei Liu, et al. 2024. Llava-onevision: Easy visual task transfer. *arXiv preprint arXiv:2408.03326* (2024).

[21] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. 2023. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *International conference on machine learning*. PMLR, 19730–19742.

[22] Kevin Y. Li, Sachin Goyal, Joao D. Semedo, and J. Zico Kolter. 2024. Inference Optimal VLMs Need Only One Visual Token but Larger Models. arXiv:2411.03312 [cs.CV] https://arxiv.org/abs/2411.03312

[23] Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190* (2021).

[24] Yifan Li, Yifan Du, Kun Zhou, Jinpeng Wang, Wayne Xin Zhao, and Ji-Rong Wen. 2023. Evaluating object hallucination in large vision-language models. *arXiv preprint arXiv:2305.10355* (2023).

[25] Youwei Liang, Chongjian Ge, Zhan Tong, Yibing Song, Jue Wang, and Pengtao Xie. 2022. Not all patches are what you need: Expediting vision transformers via token reorganizations. *arXiv preprint arXiv:2202.07800* (2022).

[26] Chenfei Liao, Wensong Wang, Zichen Wen, Xu Zheng, Yiyu Wang, Haocong He, Yuanhuiyi Lyu, Lutao Jiang, Xin Zou, Yuqian Fu, et al. 2025. Are we using the right benchmark: An evaluation framework for visual token compression methods. *arXiv preprint arXiv:2510.07143* (2025).

[27] Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. 2024. Improved baselines with visual instruction tuning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 26296–26306.

[28] Haotian Liu, Chunyuan Li, Yuheng Li, Bo Li, Yuanhan Zhang, Sheng Shen, and Yong Jae Lee. 2024. Llavanext: Improved reasoning, ocr, and world knowledge.

[29] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2023. Visual instruction tuning. *Advances in neural information processing systems* 36 (2023), 34892–34916.

[30] Yuan Liu, Haodong Duan, Yuanhan Zhang, Bo Li, Songyang Zhang, Wangbo Zhao, Yike Yuan, Jiaqi Wang, Conghui He, Ziwei Liu, et al. 2024. Mmbench: Is your multi-modal model an all-around player?. In *European conference on computer vision*. Springer, 216–233.

[31] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems* 35 (2022), 27730–27744.

[32] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. In *International conference on machine learning*. PmLR, 8748–8763.

[33] Yongming Rao, Wenliang Zhao, Benlin Liu, Jiwen Lu, Jie Zhou, and Cho-Jui Hsieh. 2021. Dynamicvit: Efficient vision transformers with dynamic token sparsification. *Advances in neural information processing systems* 34 (2021), 13937–13949.

[34] Anna Rogers, Olga Kovaleva, and Anna Rumshisky. 2020. A primer in BERTology: What we know about how BERT works. *Transactions of the association for computational linguistics* 8 (2020), 842–866.

[35] Yuzhang Shang, Mu Cai, Bingxin Xu, Yong Jae Lee, and Yan Yan. 2025. Llava-prumerge: Adaptive token reduction for efficient large multimodal models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 22857–22867.

[36] Zhenwei Shao, Mingyang Wang, Zhou Yu, Wenwen Pan, Yan Yang, Tao Wei, Hongyuan Zhang, Ning Mao, Wei Chen, and Jun Yu. 2025. Growing a twig to accelerate large vision-language models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 20064–20074.

[37] Shengbang Tong, Zhuang Liu, Yuexiang Zhai, Yi Ma, Yann LeCun, and Saining Xie. 2024. Eyes wide shut? exploring the visual shortcomings of multimodal llms. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 9568–9578.

[38] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971* (2023).

[39] Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, et al. 2024. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191* (2024).

[40] Zichen Wen, Yifeng Gao, Shaobo Wang, Junyuan Zhang, Qintong Zhang, Weijia Li, Conghui He, and Linfeng Zhang. 2025. Stop looking for important tokens in multimodal language models: Duplication matters more. *arXiv preprint arXiv:2502.11494* (2025).

[41] Senqiao Yang, Yukang Chen, Zhuotao Tian, Chengyao Wang, Jingyao Li, Bei Yu, and Jiaya Jia. 2025. Visionzip: Longer is better but not necessary in vision language models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*. 19792–19802.

[42] Yuan Yao, Tianyu Yu, Ao Zhang, Chongyi Wang, Junbo Cui, Hongji Zhu, Tianchi Cai, Haoyu Li, Weilin Zhao, Zhihui He, et al. 2024. Minicpm-v: A gpt-4v level mllm on your phone. *arXiv preprint arXiv:2408.01800* (2024).

[43] Xubing Ye, Yukang Gan, Yixiao Ge, Xiao-Ping Zhang, and Yansong Tang. 2025. Atp-llava: Adaptive token pruning for large vision language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 24972–24982.

[44] Xubing Ye, Yukang Gan, Xiaoke Huang, Yixiao Ge, and Yansong Tang. 2025. Voco-llama: Towards vision compression with large language models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*. 29836–29846.

[45] Weihao Yu, Zhengyuan Yang, Linjie Li, Jianfeng Wang, Kevin Lin, Zicheng Liu, Xinchao Wang, and Lijuan Wang. 2023. Mm-vet: Evaluating large multimodal models for integrated capabilities. *arXiv preprint arXiv:2308.02490* (2023).

[46] Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. 2023. Sigmoid loss for language image pre-training. In *Proceedings of the IEEE/CVF international conference on computer vision*. 11975–11986.

[47] Qizhe Zhang, Aosong Cheng, Ming Lu, Renrui Zhang, Zhiyong Zhuo, Jiajun Cao, Shaobo Guo, Qi She, and Shanghang Zhang. 2025. Beyond text-visual attention: Exploiting visual cues for effective token pruning in vlms. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 20857–20867.

[48] Yuan Zhang, Chun-Kai Fan, Junpeng Ma, Wenzhao Zheng, Tao Huang, Kuan Cheng, Denis Gudovskiy, Tomoyuki Okuno, Yohei Nakata, Kurt Keutzer, et al. 2024. Sparsevlm: Visual token sparsification for efficient vision-language model inference. *arXiv preprint arXiv:2410.04417* (2024).