

Layer-wise Vision Injection with Disentangled Attention for Efficient LVLMs

Xuange Zhang^{1†}, Dengjie Li^{2†}, Bo Liu¹, Zenghao Bao², Yao Zhou², Baisong Yang²,
Zhongying Liu², Yujie Zhong^{2*}, Tongtong Yuan^{1*}

¹Beijing University of Technology, CN ²Meituan Inc., CN

xuange020@163.com, {liubo, yuantt}@bjut.edu.cn,

{lidengjie, baozenghao, zhouyao11, yangbaisong02, liuzhongying, zhongyujie}@meituan.com

Abstract

Benefiting from recent advancements in large language models and modality alignment techniques, existing Large Vision-Language Models (LVLMs) have achieved prominent performance across a wide range of scenarios. However, the excessive computational complexity limits the widespread use of these models in practical applications. We argue that one main bottleneck in computational complexity is caused by the involvement of redundant vision sequences in model computation. This is inspired by a reassessment of the efficiency of vision and language information transmission in the language decoder of LVLMs. Then, we propose a novel vision-language interaction mechanism called *Layer-wise Vision Injection with Disentangled Attention (LVIDA)*. In LVIDA, only the language sequence undergoes full forward propagation, while the vision sequence interacts with the language at specific stages within each language decoder layer. It is striking that our approach significantly reduces computational complexity with minimal performance loss. Specifically, LVIDA achieves approximately a 10× reduction in the computational cost of the language decoder across multiple LVLM models while maintaining comparable performance. Project Page: <https://xuange923.github.io/LVIDA/>

1. Introduction

Understanding complex content is a crucial step toward achieving Artificial General Intelligence (AGI). As two fundamental modalities for information processing, vision and language exhibit distinct advantages: **text** provides detailed semantic information, while **images** offer intuitive visual cues. Effectively integrating information from these two modalities has become a significant focus in current research. Although existing Large Vision-Language Models (LVLMs) [3, 15, 18, 22, 23, 28, 46] have demonstrated re-

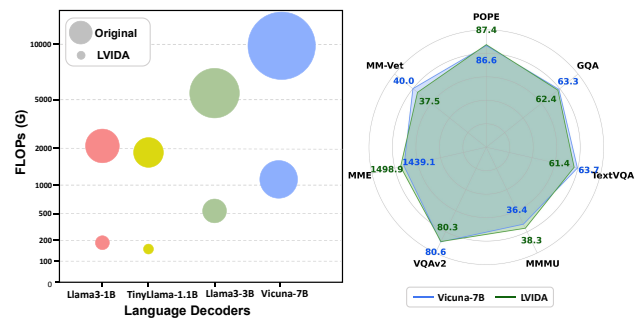


Figure 1. **Comprehensive Comparison of LVIDA against Baseline LVLMs Across Different LLM Decoders.** **Left:** FLOPs comparison across different models, showing 90% computation reduction for models under 3B and 88% for 7B. **Right:** Performance comparison across 7 benchmarks, demonstrating capability of LVIDA preservation compared to original models.

markable progress in multimodal semantic understanding, they face a critical challenge of prohibitive computational overhead caused by cross-modal interactions. The computational complexity of the self-attention increases quadratically with the sequence length, and visual sequences are often much longer than textual ones. For instance, SigLip [42] generates visual feature sequences of length 728 per image, while TextVQA datasets exhibit an average text length of only 60 tokens. This gap further amplifies when processing high-resolution images or video sequences [16, 21, 33], posing significant deployment challenges in resource-constrained scenarios. Although employing smaller language models [14, 45, 48] can reduce computation costs, this approach risks compromising model capability.

Previous research primarily focuses on the compression or elimination of visual tokens [10, 11, 44, 47]. Some methods [8, 29, 32] compress visual sequences before feeding them to a Large Language Models (LLMs). Alternative methods propose reducing visual tokens at specific LVLM layers [12, 26, 38]. These methods often require manually specifying early visual exiting layers or determining a suit-

† Equal Contribution. * Corresponding author.

able visual token pruning ratio, reducing flexibility across different task scenarios. A common issue with these approaches is that prematurely removing visual tokens in the shallow layers of LVLMs or applying an excessively high pruning ratio can hinder the model’s ability to comprehend visual information, leading to performance degradation. On the other hand, delaying token removal or using a lower pruning ratio results in only a limited reduction in computational overhead. Overall, such visual token compression methods offer limited improvements in computational efficiency while maintaining model accuracy, as they inevitably lead to the loss of critical visual information in LVLMs.

To address these issues, we revisit the vision-language interaction paradigm of LVLMs and propose a novel layer-wise disentangled attention mechanism. This approach significantly reduces computational costs while preserving model performance. It is worth noting that the long sequences composed of visual and textual features undergo extensive self-attention computations and FFN forward propagation in each layer of the LVLM decoder, resulting in significant computational overhead. If we can transform this large-scale interaction of the long sequences into a smaller-scale interaction between the visual and textual sequences, it will substantially reduce computational costs by avoiding redundant feature computations.

Inspired by these insights, we propose **Layer-wise Vision Injection with Disentangled Attention (LVIDA)**. The core innovation lies in decoupling the vision-language interaction within LVLMs: in the decoder, only the forward propagation of the language sequence is retained, while visual sequences participate in cross-modal interaction solely through attention mechanisms. This approach eliminates the layer-by-layer propagation of visual sequences within the decoder, significantly reducing computational overhead. Recognizing that single-layer interaction may prevent the model from fully absorbing visual information, we introduce a layer-wise multiple injection strategy. This replaces the original large-scale interactions with multiple smaller-scale interactions at each layer, effectively reducing computational costs while minimizing performance degradation. As shown in Figure 1, LVIDA **reduces the computational cost of multiple LVLM decoders by 90%** while maintaining comparable model performance. Our primary contributions include:

- Revealing and analyzing the phenomenon of redundant transmission of visual features in the LVLM decoder from a propagation perspective, and proposing a disentangled vision-language interaction paradigm for LVLMs.
- Introducing the Layer-wise Vision Injection with Disentangled Attention (LVIDA) method, which achieves efficient vision-language interaction through disentangled attention and a hierarchical multi-injection strategy.
- Through extensive experiments, LVIDA has been proven

to significantly reduce computational overhead while maintaining strong performance across various LVLMs. Compared to state-of-the-art visual token reduction methods, LVIDA demonstrates superior performance. Furthermore, comprehensive ablation studies validate the effectiveness of each module within our LVIDA.

2. Related Works

2.1. Large Vision-Language Models

With the remarkable performance of LLMs [1, 4, 35, 36, 41, 43] in natural language processing, researchers have extended these models to the multimodal domain, enabling more advanced vision and language understanding tasks. The typical architecture of LVLMs [5, 13, 28, 45, 46] includes three main components: a visual encoder, a vision-language connector, and a large language model. Most approaches focus on aligning visual features with the textual space effectively. In this field, Flamingo [2] pioneers the use of resamplers and cross-attention dense blocks to integrate visual features into LLMs. Following this, BLIP-2 [23] introduces Qformer, which leverages frozen image encoders and LLMs to perform various vision-language tasks. The LLaVA series [21, 27, 28] and MiniGPT-4 [46] apply simple but effective projection layers to align different modalities and improved instruction-following abilities through multimodal instruction tuning data. Fuyu [6] aligns image patches to the LLM embeddings through a simple linear mapping. InternVL [13] introduces a large-scale vision encoder that integrates with QLLaMA as middleware for alignment with the LLM. Qwen-VL [5] proposes a Position-aware Vision-Language Adapter, which compresses visual feature sequences to a fixed length before input to the LLM. However, as model sizes continue to increase, the computational resources required also grow, calling for more efficient input processing methods to manage the increasing contextual demands.

2.2. Vision Token Reduction

The primary reason for the high computational cost of LVLMs is the excessive length of input sequences, often due to numerous prefix visual tokens. As computational costs scale quadratically with the number of input tokens, researchers have explored methods to reduce visual tokens and thereby lower computational expenses effectively.

FastV [12] is among the first to identify the inefficiencies associated with visual token usage in LVLMs. This model uses signals from the LLM to guide visual token pruning, trimming half of the image tokens within intermediate layers of the LLM. VTW [25] takes this further by completely removing all visual tokens after a certain intermediate layer, relying only on text tokens for subsequent computations. LLaVA-PruMerge [31] dynamically retains visual tokens

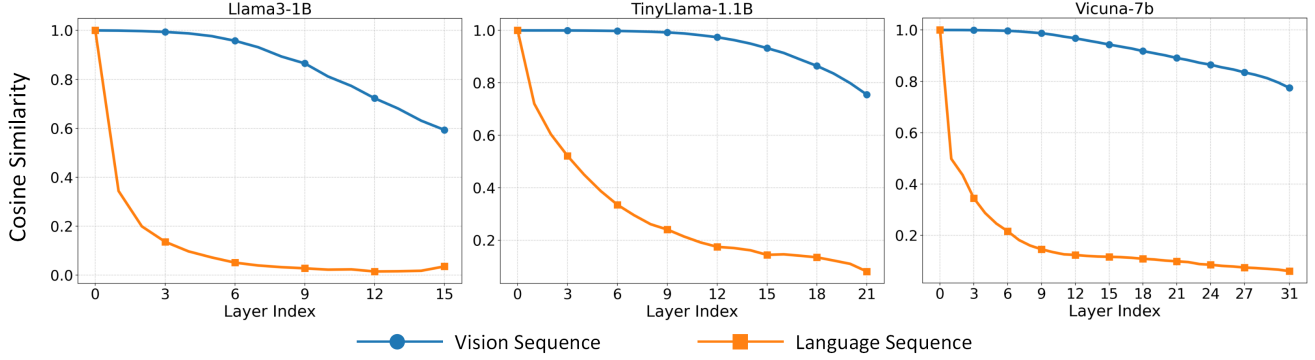


Figure 2. Layer-wise Cosine Similarity of Vision and Language Sequences across different language decoder.

based on their similarity, effectively compressing the number of visual tokens. M^3 [9], inspired by Matryoshka Representation Learning, learns multi-granular, coarse-to-fine token representations to control the number of tokens involved in computation.

While these approaches effectively reduce computational loads, they primarily focus on pruning or fully excluding visual tokens from certain computations. However, these methods will potentially impair the LVLMs’ performance due to the loss of critical visual information. To avoid visual information loss, we develop a novel disentangled attention module to adjust the propagation of vision sequences in the LVLM decoder rather than reduce vision tokens. This approach can significantly reduce computational cost while preserving model performance.

3. Method

In Section 3.1, we introduce the implementation details of commonly used LVLMs. Section 3.2 provides a detailed analysis of Redundant Vision Propagation in the LVLM decoder. Finally, in Section 3.3, we present the implementation details and algorithmic workflow of LVIDA.

3.1. Preliminary

We briefly introduce the commonly used LVLM architecture, referred to as Vanilla-LVLM. The framework comprises three core components: a vision encoder, a connector, and a language decoder. Given an input image I , the vision encoder extracts visual features: $\mathbf{X}_v = f_{VE}(I) \in \mathbb{R}^{n \times d_v}$, where n denotes the sequence length and d_v represents the feature dimension. These visual features are then processed through a connector to align with language features: $\mathbf{X}_{v'} = f_{Connector}(\mathbf{X}_v) \in \mathbb{R}^{n \times d}$. For textual input T , a text encoder within the LLM extracts its feature representation: $\mathbf{X}_l = f_{TE}(T) \in \mathbb{R}^{m \times d}$, where m indicates the length of the language sequence. The vision and text feature sequences are then concatenated to form the multimodal input:

$$\mathbf{X}_{vl} = [\mathbf{X}_{v'}; \mathbf{X}_l] \quad (1)$$

The language decoder is a Transformer model with L layers, generating text autoregressively. Each layer consists of two sub-layers: self-attention and feed-forward network (FFN). For layer i ($1 \leq i \leq L$), the attention layer includes three linear transformation matrices, \mathbf{W}^Q , \mathbf{W}^K , and \mathbf{W}^V , which transform the input \mathbf{X}_{vl} into \mathbf{Q} , \mathbf{K} , and \mathbf{V} matrices. The attention output is computed as:

$$\mathbf{H}_a = \text{Softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d}}\right)\mathbf{V} \quad (2)$$

where \mathbf{H}_a represents the hidden state after the attention computation. A causal mask prevents information leakage from future tokens during decoding. The FFN processes the attention output through two linear transformations with intermediate activation:

$$\mathbf{Y} = \mathbf{W}_2(\text{Act}(\mathbf{W}_1\mathbf{H}_a)) \quad (3)$$

where \mathbf{W}_1 and \mathbf{W}_2 are the weight matrices for linear transformations. \mathbf{Y} represents the output of decoder layer. After L layers, the decoder generates a probability distribution over the next text token. The Vanilla-LVLM architecture is illustrated on the left side of Figure 3.

For an input sequence of length $n + m$, the computational complexity of the Vanilla Model includes two parts: self-attention requires $O((n + m)^2d)$ due to the $\mathbf{Q}\mathbf{K}^\top$ operation, while the feed-forward network, with two linear layers, contributes $O(8(n + m)d^2)$. Thus, the total complexity is:

$$O((n + m)^2d) + O(8(n + m)d^2) \quad (4)$$

Since the vision sequence length n is typically longer than the language sequence length m , reducing visual tokens can significantly lower computational cost.

3.2. Analysis on Redundant Vision Propagation

We conduct a hierarchical analysis to examine the representational differences between visual and language sequences during the forward propagation of the LVLM decoder. To quantify variations between the original input

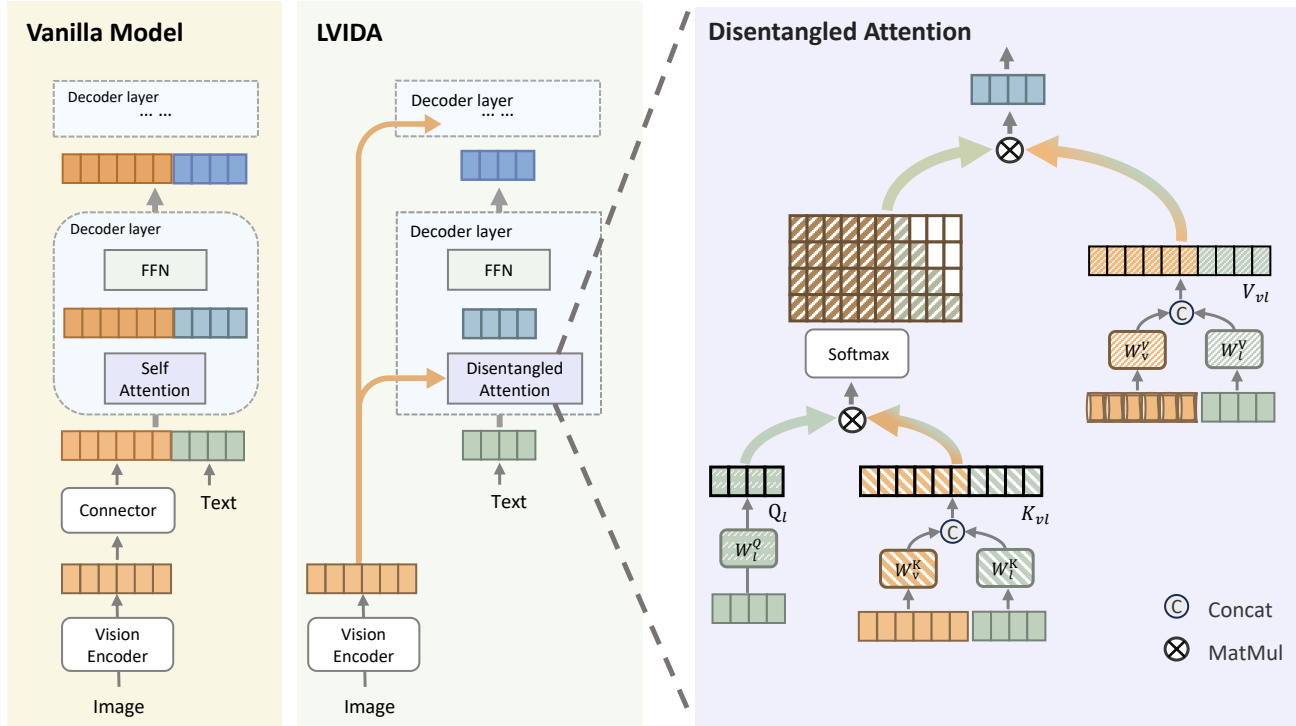


Figure 3. Comparison of Vanilla Model and LVIDA Architectures. **Left:** Overall structure of traditional Vanilla. **Middle:** Overall structure of LVIDA. **Right:** Details of Disentangled Attention.

and layer-wise output representations, we use cosine similarity to measure directional consistency between vectors. Given output sequences \mathbf{Y}^i from decoder layer i , we partition them into a visual part \mathbf{Y}_v^i and textual part \mathbf{Y}_t^i . To ensure a fair comparison of variations across layers, we compute the token-wise similarity for each representation and take the average.

For the visual part, we define a visual sequence at layer i as $\mathbf{Y}_v^i = \{\mathbf{Y}_{v,1}^i, \mathbf{Y}_{v,2}^i, \dots, \mathbf{Y}_{v,n}^i\}$. The layer-wise visual divergence is measured as:

$$S_v(i) = \frac{1}{n} \sum_{j=1}^n \cos(\mathbf{Y}_{v,j}^0, \mathbf{Y}_{v,j}^i) \quad (5)$$

Values closer to 1 indicates that the semantic representation of the entire image at layer i undergoes minimal change compared to the original input.

For the textual part, we define a textual sequence at layer i as: $\mathbf{Y}_t^i = \{\mathbf{Y}_{1,1}^i, \mathbf{Y}_{1,2}^i, \dots, \mathbf{Y}_{1,m}^i\}$. The layer-wise textual divergence is computed as:

$$S_t(i) = \frac{1}{m} \sum_{k=1}^m \cos(\mathbf{Y}_{1,k}^0, \mathbf{Y}_{1,k}^i) \quad (6)$$

Lower values indicate larger semantic transformation of textual features as the network depth increases. We ana-

lyze the evolution of visual and textual features across different layers using various LLMs as decoders, with results shown in Figure 2. To ensure a pure representation comparison, we exclude the final decoder layer from our analysis. Across all models, the visual sequence consistently maintains a high similarity to the original input at every layer. In contrast, the textual sequence undergoes significant transformations as depth increases. This observation suggests that even in LVLm decoders, computations primarily revolve around textual part. Since visual sequences have already been transitioned to high-level semantic information during the encoding phase, propagating them through all decoder layers alongside text is redundant. This observation can strongly justify our proposition to eliminate full-layer visual sequence propagation with a disentangled vision-language interaction paradigm for LVLms.

3.3. LVIDA

Inspired by the observation of redundant vision propagation, we introduce **Layer-wise Vision Injection with Disentangled Attention (LVIDA)**. The core idea is to disentangle the interaction between vision and language at each layer of the decoder: the attention sublayer is responsible for multimodal fusion, while the FFN sublayer processes only textual information. This approach ensures effective

multimodal integration while significantly reducing computational costs. The middle part of Figure 3 illustrates the overall structure of LVIDA, while the right side shows the detailed implementation of Mixture Attention.

3.3.1. Disentangled Attention

In Disentangled Attention, the vision sequence \mathbf{X}_v and language sequence \mathbf{X}_l undergo separate projections. Vision sequence are transformed using vision projection matrices \mathbf{W}_v^K and \mathbf{W}_v^V , producing \mathbf{K}_v and \mathbf{V}_v . Language sequence are transformed via \mathbf{W}_l^Q , \mathbf{W}_l^K , and \mathbf{W}_l^V to produce matrices \mathbf{Q}_l , \mathbf{K}_l , and \mathbf{V}_l .

To integrate multimodal information, we concatenate the vision and language key-value matrices:

$$\mathbf{K}_{vl} = [\mathbf{K}_v; \mathbf{K}_l], \quad \mathbf{V}_{vl} = [\mathbf{V}_v; \mathbf{V}_l] \quad (7)$$

Multimodal interaction is then achieved through attention computation:

$$\mathbf{H}_a^i = \text{Softmax} \left(\frac{\mathbf{Q}_l \mathbf{K}_{vl}^\top}{\sqrt{d}} \right) \mathbf{V}_{vl} \quad (8)$$

During attention computation, positional embeddings are added to \mathbf{Q}_l and \mathbf{K}_l only, as visual features retain positional information from encoding. To maintain autoregressive behavior, we apply a partial causal mask to the attention weights: All language tokens can attend to previous visual tokens. Language-to-language attention follows a causal mask to prevent future information leakage. The language sequence then passes through the FFN of the current layer, and after processing through L layers, the final output is obtained.

For an input sequence of length $n + m$, the computational complexity of LVIDA includes two parts: Disentangled Attention requires $O((n + m)md)$ due to the $\mathbf{Q}_l \mathbf{K}_{vl}^\top$ operation, while the feed-forward network, applied only to the language sequence, contributes $O(8md^2)$. Thus, the total complexity is:

$$O((n + m)md) + O(8md^2) \quad (9)$$

Compared to the complexity of the Vanilla-LVLM, LVIDA significantly reduces computational cost. By avoiding the quadratic term associated with self-attention across both sequences, LVIDA provides a more efficient solution, especially advantageous for handling longer vision sequences.

3.3.2. Layer-Wise Vision Injection

After replacing Self-Attention with Disentangled Attention, the vision sequence is no longer propagated across layers. However, relying on a single-layer visual interaction may be insufficient for capturing rich visual features. To address this, we propose Layer-Wise Vision Injection, where visual information is injected into each decoder layer. We integrate the original LVLM connector with projection matrices \mathbf{W}_v^K and \mathbf{W}_v^V . These projection layers align feature

dimensions across modalities, ensuring smooth integration between vision and language representations. Meanwhile, visual information is accessible at each layer without interrupting the propagation of the language sequence.

The complete algorithmic workflow of LVIDA is presented in Algorithm 1.

Algorithm 1 LVIDA: Layer-wise Vision Injection with Disentangled Attention

```

1: Input: Image  $I$ , Text  $T$ 
2: Output: Language sequence  $\mathbf{Y}^L$  after  $L$  layers of processing
3: Initialize:
4:    $\mathbf{X}_v \leftarrow f_{\text{VE}}(I)$ 
5:    $\mathbf{X}_l \leftarrow f_{\text{TE}}(T)$ 
6:    $\mathbf{Y}^0 \leftarrow \mathbf{X}_l$ 
7: for  $i = 1$  to  $L$  do
8:   Language Projection:
9:    $\mathbf{Q}_l^i, \mathbf{K}_l^i, \mathbf{V}_l^i \leftarrow \text{Proj}_l(\mathbf{Y}^{i-1})$ 
10:  Vision Projection:
11:   $\mathbf{K}_v^i, \mathbf{V}_v^i \leftarrow \text{Proj}_v(\mathbf{X}_v)$ 
12:  Add Positional Embedding:
13:   $\mathbf{Q}_{l'}^i \leftarrow \mathbf{Q}_l^i + \mathbf{P}_l, \mathbf{K}_{l'}^i \leftarrow \mathbf{K}_l^i + \mathbf{P}_l$ 
14:  Concatenate Sequences:
15:   $\mathbf{K}_{vl}^i \leftarrow [\mathbf{K}_v^i; \mathbf{K}_{l'}^i], \mathbf{V}_{vl}^i \leftarrow [\mathbf{V}_v^i; \mathbf{V}_l^i]$ 
16:  Calculate Attention Scores:
17:   $\mathbf{S}_a^i \leftarrow (\mathbf{Q}_{l'}^i (\mathbf{K}_{vl}^i)^\top) / \sqrt{d}$ 
18:  Apply Part-Causal Mask:
19:   $\hat{\mathbf{S}}_a^i \leftarrow \mathbf{S}_a^i + \text{Mask}$ 
20:  Compute Attention Output:
21:   $\mathbf{H}_a^i \leftarrow \text{Softmax}(\hat{\mathbf{S}}_a^i) \mathbf{V}_{vl}^i$ 
22:  Feed-forward Network:
23:   $\mathbf{Y}^i \leftarrow \mathbf{W}_2(\text{Act}(\mathbf{W}_1 \mathbf{H}_a^i))$ 
24: end for
25: return  $\mathbf{Y}^L$ 

```

4. Experiments

To validate the effectiveness of the proposed LVIDA method, we conduct extensive experiments and performance analyses. In Section 4.2, we select large language models (LLMs) of varying scales, ranging from 1B to 7B parameters, as decoders for LVLMs and comprehensively evaluate the original models alongside their LVIDA-enhanced counterparts. Subsequently, in Section 4.3, we modify the LLaVA model with LVIDA, compare it against mainstream LVLM models, and assess the performance of different inference acceleration methods within the same framework. In Section 4.4, we test smaller-scale vision encoders under the same decoder to explore model performance when overall computational load is minimized. In Section 4.5, we perform ablation studies to thoroughly eval-

uate the proposed method, further demonstrating the superiority of the LVIDA model.

4.1. Setup

Model Selection. We select two different scales of LLMs as language decoders. **Small-scale:** TinyLlama-1.1B, Llama3.2-1B, Llama3.2-3B and **Normal-scale:** Vicuna-7B. For the visual encoder, we adopt a pre-trained SigLIP [42] SoViT-400M/14, which generates a visual sequence of length 728 per image. In the vanilla-LVLM setup, following the configuration of TinyLLaVA [45], the connector is implemented as a 2-layer MLP.

Evaluation Metrics. To comprehensively assess our LVIDA method, we evaluate both model performance and computational efficiency.

For model performance, we conduct evaluations on nine widely used benchmarks: VQAv2 [19], GQA [20], TextVQA [34], MM-Vet [39], POPE [24], MME [17], MMMU [40], OCRVQA [30] and STVQA [7]. These benchmarks collectively assess the model’s capabilities, ranging from basic visual perception to advanced reasoning.

Computational efficiency is quantified using three metrics: (1) **FLOPs:** Computation cost under varying vision-language input ratios (2) **TTFT:** Time to First Token reflecting real-time responsiveness (3) **Peak Memory:** Maximum GPU memory consumption during inference

Implementation Details To accommodate different scales of language decoders, we design tailored training strategies. For small-scale models, pretraining is conducted on the LLaVA-1.5-558K dataset, keeping both the visual encoder and language decoder frozen while optimizing only the visual projection layer. During fine-tuning, we use the LLaVA-1.5-mix-665K dataset, freezing the visual encoder while updating all parameters of the language decoder.

For normal-scale models, training data is sourced from LLaVA-OneVision. The pretraining phase utilizes Stage 1.5 with approximately 4M samples, while the fine-tuning phase employs Stage 2-SI with around 3.2M samples for instruction tuning. The update strategy remains the same as in small-scale models.

4.2. Comprehensive Assessment

We use LVLMs with the original LLM as decoders as baselines. We compare these baseline models to LVIDA, focusing on computational efficiency and performance accuracy. To thoroughly assess the computational efficiency improvements introduced by LVIDA, we evaluate its performance against the baseline models across various visual-language input lengths. Since enhancements in LVIDA target the language decoder within LVLMs, our reported computational metrics specifically reflect the decoder’s performance.

Evaluation Across Decoders. Table 1 presents the comprehensive evaluation of LVIDA and baseline models across different scales of language decoders. To simulate typical multimodal input scenarios, we set the text token length to 64. The vision-language input ratio (V-L input ratio) is 728:64. The results demonstrate that LVIDA significantly reduces computational cost across all models. For small-scale models, LVIDA achieves a **90% reduction** in decoder FLOPs, while for the 7B model, computational cost is **reduced by 88%**. In terms of real-time responsiveness, TTFT reveals progressive acceleration with model scaling. The 1B model achieves a **2.3× speedup**, while the 7B model attains a **4× acceleration**. Regarding hardware resource consumption, LVIDA consistently **reduces peak memory consumption**. While achieving lower computation costs, faster response times, and reduced memory requirements, LVIDA maintains comparable performance to the original models on most benchmark tests. This also confirms the effectiveness of LVIDA in balancing efficiency and accuracy.

Evaluation Under Varying V-L Input Ratios. Table 2 reports the computational efficiency of LVIDA under varying vision-language input ratios. When the language sequence is significantly shorter than the visual sequence, LVIDA exhibits exceptional computational optimization. The 1B model reduces FLOPs by 94%, with a 2.4× speedup in TTFT, while the 7B model achieves a 92% reduction in GFLOPs and a 5.2× speedup in TTFT. Even as language sequence length increases, LVIDA consistently maintains a clear computational cost advantage, making it well-suited for both general applications and complex tasks.

4.3. Comparative Experiments

Cross-Method Performance Comparison As shown in Table 3, we compare LVIDA against two categories of baselines: mainstream LVLMs and visual token compression methods, alongside our baseline and method. Despite architectural and training data scale differences among compared methods, our approach achieves performance levels comparable to most existing works.

Homogeneous Framework Comparison. For direct comparison, we implement FastV [12] and VTW [26] on Llama3-1B and Vicuna-7B baselines, with results detailed in Table 4. The FastV setup removes visual tokens after layer K with retention ratio R, while the VTW setup discards all visual tokens after layer K. Although both methods remove a portion of visual tokens, the visual sequence still participates in forward propagation in some decoder layers. In terms of FLOPs, LVIDA maintains a significant computational advantage over these baselines. Performance-wise, both FastV and VTW exhibit notable degradation on several benchmarks, indicating that excessively early or aggressive visual token compression inevitably leads to information loss. In contrast, LVIDA effectively reduces com-

Table 1. **Comprehensive Comparison of LVIDA and Baseline Models.** The V-L input ratio in these LVLMs is 728:64. The table reports computational efficiency (measured by FLOPs, Time to First Token, Peak Memory consumption) and performance, highlighting the efficiency of LVIDA with comparable results.

Language Decoder	FLOPs (G)	TTFT (ms)	Memory (GB)	VQAv2	GQA	TextVQA	MM-Vet	POPE	MME	MMM	OCRVQA	STVQA
TinyLlama-1.1B	1750	58.1	9.4	75.5	58.6	49.6	23.5	86.3	1256.5	28.3	50.1	37.3
LVIDA	161(↓90.8%)	29.1(↑2.0x)	5.5(-3.9)	75.5	59.2	44.1	24.5	85.7	1179	29.0	54.6	36.7
Llama-3.2-1B	2040	54.9	9.1	76.8	59.6	52.7	27.8	86.7	1334.5	30.6	50.2	39.5
LVIDA	192(↓90.6%)	23.4(↑2.4x)	5.6(-3.5)	76.2	59.6	50.6	28.4	86.6	1200.9	29.9	55.0	37.7
Llama-3.2-3B	5310	119.8	15.2	78.6	62.5	55.3	31.5	86.9	1449.3	34.8	55.0	39.8
LVIDA	525(↓90.1%)	37.6(↑3.2x)	9.8(-5.4)	78.4	61.4	54.4	30.2	86.6	1261.9	35.6	53.1	40.3
Vicuna-7B	10800	215.6	24.3	80.6	63.3	63.7	40.0	86.6	1439.1	36.4	58.2	50.6
LVIDA	1310(↓87.9%)	54.4(↑4.0x)	17.2(-7.1)	80.3	62.4	61.4	37.5	87.4	1498.9	38.3	55.8	53.7

Table 2. **Computational Efficiency of LVIDA and Baseline Models Under Different Vision-Language input Ratios.** V:L means the length of the Vision:language in the input sequence.

Language Decoder	V:L 728:32			V:L 728:200		
	FLOPs (G)	TTFT (ms)	Memory (GB)	FLOPs (G)	TTFT (ms)	Memory (GB)
TinyLlama-1.1B	1680	55.6	9.2	2080	73.1	10.7
LVIDA	92	28.1	5.3	466	34.8	6.4
Llama-3.2-1B	1950	53.2	8.8	2410	66.7	10.1
LVIDA	110	22.1	5.4	546	25.1	6.4
Llama-3.2-3B	5080	108.5	14.9	6260	143.1	16.8
LVIDA	310	35.9	9.6	1450	49.9	11.1
Vicuna-7B	10350	200.7	23.9	12720	250.5	26.7
LVIDA	875	38.8	16.9	3180	82.7	19.0

Table 3. **Cross-Method Performance Comparison on Multimodal Benchmarks.** The table compares mainstream LVLMs, vision compression methods, and LVIDA. Accuracy scores demonstrate that LVIDA achieves performance comparable to leading approaches.

Method	Language Decoder	VQAv2	GQA	TextVQA	MM-Vet	POPE	MME
InstructBLIP	Vicuna-7B	-	49.2	50.1	26.2	-	1084.0
Qwen-VL	Qwen-7B	-	59.3	63.8	-	-	1487.6
LLaVA-1.5	Vicuna-7B	78.5	62.0	58.2	31.1	85.9	1510.7
LLaVA-Prunmerge	Vicuna-7B	72.0	51.6	56.0	21.1	76.3	1350.3
LLaVA-Prunmerge+	Vicuna-7B	76.8	56.4	57.1	25.0	84.0	1462.4
Trim	Vicuna-7B	76.4	61.4	53.7	28.0	85.3	1461.3
FastV	Vicuna-7B	-	61.0	58.4	-	85.2	-
PyramidDrop	Vicuna-7B	-	61.9	58.5	-	86.0	-
LLaVA-1.5-SigLIP	Vicuna-7B	80.6	63.3	63.7	40.0	86.6	1439.1
LVIDA	Vicuna-7B	80.3	62.4	61.4	37.5	87.4	1498.9

computational cost while maintaining more comprehensive and superior performance. These results also indicate that our LVIDA can achieve more excellent computation efficiency with better multimodal understanding performance.

4.4. Extended Evaluation

By adopting LVIDA, the computational cost of the language decoder significantly decreases, with the vision encoder ac-

Table 4. **Homogeneous Framework Efficiency and Accuracy Tradeoffs.** GFLOPs and task accuracy under different visual token compression strategies vs. LVIDA. K denotes the layer where tokens are reduced, and R represents the reduction ratio.

Language Decoder	GFLOPs	GQA	TextVQA	MM-Vet	POPE	MMM
Llama3-1B	2040	59.6	52.7	27.8	86.7	30.6
w.FastV(K=3,R=50%)	1230	58.5	49.7	27.7	87.0	30.8
w.FastV(K=3,R=75%)	829	56.3	48.4	24.5	84.1	29.8
w.VTW(K=8)	875	47.0	38.7	16.7	83.6	29.9
w.LVIDA	192	59.6	50.6	28.4	86.6	30.4
Vicuna-7B	10800	63.3	63.7	40.0	86.6	36.4
w.FastV(K=3,R=50%)	6220	62.6	62.7	39.3	86.3	36.8
w.FastV(K=3,R=75%)	3970	58.8	58.1	34.6	82.3	37.2
w.VTW(K=16)	5710	59.2	52.7	29.2	86.3	36.3
w.LVIDA	1310	62.4	61.4	37.5	87.4	38.3

counts for the majority of the overall computational cost. To further reduce this cost, we explore the use of smaller vision encoders to evaluate model performance under minimized computational demands. Specifically, we compare the performance of TinyCLIP series [37] and Siglip as vision encoders, using TinyLlama-1.1B as the language decoder. The results are shown in Table 5.

For TinyCLIP-39M and TinyCLIP-40M, While these two encoders have the same feature dimensions, TinyCLIP-39M has a smaller patch size, enabling finer-grained image processing. Results show that TinyCLIP-39M outperformed across metrics. For TinyCLIP-40M and TinyCLIP-61M, Although these encoders have the same token count, TinyCLIP-61M features a larger dimensionality. However, increasing feature dimensions did not enhance performance. Instead, patch size, which affects feature sequence length, exerts a more substantial impact.

While using a smaller vision encoder reduces overall performance, it also achieves a substantial reduction in computational cost. In scenarios with stringent constraints on computational resources, such as deployment on edge/mobile devices, LVIDA with a smaller vision encoder can greatly facilitate the feasibility of deploying LVLMs.

Table 5. Performance and Parameters of Various Vision Encoders. All experiments use TinyLlama-1.1B (with LVIDA applied) as the language decoder. InRes: Input Image Resolution; PatchSz: Patch Size; FeatDim: Vision Feature Dimensions.

Vision Encoder	Model Parameters				Performances					
	InRes	PatchSz	FeatDim	Params (B)	GFLOPs	VQAv2	GQA	TextVQA	POPE	MME
SigLIP	384 × 384	14 × 14	728 × 1152	0.428	670.89	75.5	59.2	44.1	85.7	1179.0
TinyCLIP-39M	224 × 224	16 × 16	196 × 512	0.038	15.04	67.1	52.9	31.1	79.2	1069.5
TinyCLIP-40M	224 × 224	32 × 32	49 × 512	0.039	3.93	64.2	51.9	30.8	77.9	1022.3
TinyCLIP-61M	224 × 224	32 × 32	49 × 640	0.061	6.10	63.7	51.5	34.4	77.6	1010.2

4.5. Ablation Studies

Table 6. Comparison of Vanilla vs. Disentangled Projection using Llama3-1B as the language decoder.

Language Decoder	GFLOPs	POPE	MMU	GQA	VQAv2	MM-vet
Vanilla Proj	2040	86.7	30.6	59.6	76.8	27.8
Disentangled Proj	192	86.6	30.4	59.6	76.2	28.4

Why Disentangled Projection? Our method separately projects visual and textual features, enabling more efficient vision-language interaction. In the Vanilla model’s decoder, each layer’s KV projection for the visual sequence matches the LLM’s hidden dimension for both inputs and outputs. By contrast, our method uses the vision encoder’s native dimensionality at the decoder input and projects to the LLM’s hidden dimension at the output. When the dimension of the vision encoder (e.g., 1152 for SigLIP) is smaller than the LLM’s hidden size (e.g., 2048 for Llama3-1B), this discrepancy further reduces computational overhead. Table 6 demonstrates that disentangled projection effectively decreases FLOPs while maintaining or even improving model performance. Therefore, we adopt disentangled attention to achieve both lower computational cost and better efficiency.

Table 7. Impact of LVIDA on per-layer Computation Reduction. FLOPs comparison of standard vs disentangled attention and FFN components in single decoder layers

Language Decoder	Attn GFlops	FFN GFlops	Language Decoder	Attn GFlops	FFN GFlops
Llama3-1B	21.8	79.7	Vicuna-7B	116.6	214.3
LVIDA	3.5(↓84%)	6.4(↓92%)	LVIDA	18.4(↓84%)	8.7(↓96%)

How Much Computation Does Disentangled Attention Save? LVIDA significantly reduces computational cost by removing the visual sequence from the decoder’s forward propagation. Table 7 records the reduction in FLOPs when replacing standard self-attention with disentangled attention in Llama3-1B and Vicuna-7B. FLOPs of attention component decrease by approximately 84%, while the FFN component, which no longer processes visual sequences, sees a 92% reduction in FLOPs for the 1B model and a 96% re-

duction for the 7B model.

Table 8. Performance Impact of Vision Injection Depth. Accuracy degradation when stopping visual token injection after layer K.

Setting	TextVqa	POPE	MME	GQA	VQAv2	MM-Vet
Baseline	52.7	86.7	1334.6	59.6	76.8	27.8
K=1	39.5	80.2	1069.1	50.3	62.2	16.1
K=2	42.0	81.8	1148.2	51.5	65.8	17.3
K=50%	41.8	82.9	1185.8	53.1	69.6	22.5

Why Layer-Wise Vision Injection? To investigate the impact of visual information on model performance, we compare the effects of different numbers of visual injection layers in Table 8. Experiments retaining only 1 layer, 2 layers, and the first 50% of layers for visual sequences. The baseline in the setting is based on the Llama3-1B. All configurations exhibit performance degradation compared to the baseline. This indicates that visual information across layers continues to play a role, and retaining visual tokens in all layers achieves more comprehensive performance.

5. Conclusion

To enhance the inference efficiency of LVLMs without compromising performance, we introduce Layer-wise Vision Injection with Disentangled Attention (LVIDA), an efficient framework for vision-language interaction in LVLMs. We analyze the redundancy in the forward propagation of visual sequences within the LVLm decoder from an information propagation perspective. By eliminating unnecessary visual processing, LVIDA significantly reduces computational load, accelerates response times, and decreases memory usage, all while preserving model performance. Extensive experimental results demonstrate that LVIDA provides substantial efficiency gains across multiple mainstream LVLm benchmarks. Notably, configurations with smaller encoders and decoders are particularly advantageous for real-time computing and power-constrained applications. Overall, LVIDA provides an efficient and practical solution for efficient multimodal information processing, showing strong potential to meet a wider range of application requirements.

Acknowledgement

This work was supported by National Natural Science Foundation of China No.62402023, 62476181.

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. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023. 2
- [2] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, et al. Flamingo: a visual language model for few-shot learning. *Advances in neural information processing systems*, 35:23716–23736, 2022. 2
- [3] Anas Awadalla, Irena Gao, Josh Gardner, Jack Hessel, Yusuf Hanafy, Wanrong Zhu, Kalyani Marathe, Yonatan Bitton, Samir Yitzhak Gadre, Shiori Sagawa, Jenia Jitsev, Simon Kornblith, Pang Wei Koh, Gabriel Ilharco, Mitchell Wortsman, and Ludwig Schmidt. Openflamingo: An open-source framework for training large autoregressive vision-language models. *ArXiv*, abs/2308.01390, 2023. 1
- [4] Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023. 2
- [5] Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. Qwen-vl: A frontier large vision-language model with versatile abilities. *ArXiv*, abs/2308.12966, 2023. 2
- [6] Rohan Bavishi, Erich Elsen, Curtis Hawthorne, Maxwell Nye, Augustus Odena, Arushi Somani, and Saġnak Taşırlar. Introducing our multimodal models, 2023. 2
- [7] Ali Furkan Biten, Ruben Tito, Andres Mafla, Lluís Gomez, Marçal Rusiñol, Ernest Valveny, C. V. Jawahar, and Dimosthenis Karatzas. Scene text visual question answering, 2019. 6
- [8] Mu Cai, Jianwei Yang, Jianfeng Gao, and Yong Jae Lee. Matryoshka multimodal models, 2024. 1
- [9] Mu Cai, Jianwei Yang, Jianfeng Gao, and Yong Jae Lee. Matryoshka multimodal models. *arXiv preprint arXiv:2405.17430*, 2024. 3
- [10] Jianjian Cao, Peng Ye, Shengze Li, Chong Yu, Yansong Tang, Jiwen Lu, and Tao Chen. Madtp: Multimodal alignment-guided dynamic token pruning for accelerating vision-language transformer, 2024. 1
- [11] Qingqing Cao, Bhargavi Paranjape, and Hannaneh Hajishirzi. Pumer: Pruning and merging tokens for efficient vision language models, 2023. 1
- [12] Liang Chen, Haozhe Zhao, Tianyu Liu, Shuai Bai, Junyang Lin, Chang Zhou, and Baobao Chang. An image is worth 1/2 tokens after layer 2: Plug-and-play inference acceleration for large vision-language models. In *European Conference on Computer Vision*, pages 19–35. Springer, 2025. 1, 2, 6
- [13] Zhe Chen, Jiannan Wu, Wenhai Wang, Weijie Su, Guo Chen, Sen Xing, Muyan Zhong, Qinglong Zhang, Xizhou Zhu, Lewei Lu, et al. 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*, pages 24185–24198, 2024. 2
- [14] Xiangxiang Chu, Limeng Qiao, Xinyang Lin, Shuang Xu, Yang Yang, Yiming Hu, Fei Wei, Xinyu Zhang, Bo Zhang, Xiaolin Wei, and Chunhua Shen. Mobilevlm : A fast, strong and open vision language assistant for mobile devices, 2023. 1
- [15] Wenliang Dai, Junnan Li, Dongxu Li, Anthony Meng Huat Tiong, Junqi Zhao, Weisheng Wang, Boyang Albert Li, Pascale Fung, and Steven C. H. Hoi. Instructblip: Towards general-purpose vision-language models with instruction tuning. *ArXiv*, abs/2305.06500, 2023. 1
- [16] Xiaoyi Dong, Pan Zhang, Yuhang Zang, Yuhang Cao, Bin Wang, Linke Ouyang, Songyang Zhang, Haodong Duan, Wenwei Zhang, Yining Li, Hang Yan, Yang Gao, Zhe Chen, Xinyue Zhang, Wei Li, Jingwen Li, Wenhai Wang, Kai Chen, Conghui He, Xingcheng Zhang, Jifeng Dai, Yu Qiao, Dahua Lin, and Jiaqi Wang. Internlm-xcomposer2-4khd: A pioneering large vision-language model handling resolutions from 336 pixels to 4k hd, 2024. 1
- [17] 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. Mme: A comprehensive evaluation benchmark for multimodal large language models. *ArXiv*, abs/2306.13394, 2023. 6
- [18] Tao Gong, Chengqi Lyu, Shilong Zhang, Yudong Wang, Miao Zheng, Qianmengke Zhao, Kuikun Liu, Wenwei Zhang, Ping Luo, and Kai Chen. Multimodal-gpt: A vision and language model for dialogue with humans. *ArXiv*, abs/2305.04790, 2023. 1
- [19] Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. Making the v in vqa matter: Elevating the role of image understanding in visual question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6904–6913, 2017. 6
- [20] Drew A Hudson and Christopher D Manning. 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*, pages 6700–6709, 2019. 6
- [21] Bo Li, Yuanhan Zhang, Dong Guo, Renrui Zhang, Feng Li, Hao Zhang, Kaichen Zhang, Yanwei Li, Ziwei Liu, and Chunyuan Li. Llava-onevision: Easy visual task transfer. *ArXiv*, abs/2408.03326, 2024. 1, 2
- [22] Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *International conference on machine learning*, pages 12888–12900. PMLR, 2022. 1
- [23] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *International conference on machine learning*, pages 19730–19742. PMLR, 2023. 1, 2

- [24] Yifan Li, Yifan Du, Kun Zhou, Jinpeng Wang, Wayne Xin Zhao, and Ji-Rong Wen. Evaluating object hallucination in large vision-language models. *arXiv preprint arXiv:2305.10355*, 2023. 6
- [25] Zhihang Lin, Mingbao Lin, Luxi Lin, and Rongrong Ji. Boosting multimodal large language models with visual tokens withdrawal for rapid inference. *arXiv preprint arXiv:2405.05803*, 2024. 2
- [26] Zhihang Lin, Mingbao Lin, Luxi Lin, and Rongrong Ji. Boosting multimodal large language models with visual tokens withdrawal for rapid inference. *arXiv preprint arXiv:2405.05803*, 2024. 1, 6
- [27] Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning. *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 26286–26296, 2023. 2
- [28] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *Advances in neural information processing systems*, 36, 2024. 1, 2
- [29] Yuliang Liu, Biao Yang, Qiang Liu, Zhang Li, Zhiyin Ma, Shuo Zhang, and Xiang Bai. Textmonkey: An ocr-free large multimodal model for understanding document, 2024. 1
- [30] Anand Mishra, Shashank Shekhar, Ajeet Kumar Singh, and Anirban Chakraborty. Ocr-vqa: Visual question answering by reading text in images. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 947–952, 2019. 6
- [31] Yuzhang Shang, Mu Cai, Bingxin Xu, Yong Jae Lee, and Yan Yan. Llava-prumerge: Adaptive token reduction for efficient large multimodal models. *arXiv preprint arXiv:2403.15388*, 2024. 2
- [32] Yuzhang Shang, Mu Cai, Bingxin Xu, Yong Jae Lee, and Yan Yan. Llava-prumerge: Adaptive token reduction for efficient large multimodal models, 2024. 1
- [33] Qi She, Junwen Pan, Xin Wan, Rui Zhang, Dawei Lu, and Kai Huang. Mammothmoda: Multi-modal large language model, 2024. 1
- [34] Amanpreet Singh, Vivek Natarajan, Meet Shah, Yu Jiang, Xinlei Chen, Dhruv Batra, Devi Parikh, and Marcus Rohrbach. Towards vqa models that can read. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8317–8326, 2019. 6
- [35] Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, et al. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*, 2024. 2
- [36] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: open and efficient foundation language models. *arxiv. arXiv preprint arXiv:2302.13971*, 2023. 2
- [37] Kan Wu, Houwen Peng, Zhenghong Zhou, Bin Xiao, Mengchen Liu, Lu Yuan, Hong Xuan, Michael Valenzuela, Xi Stephen Chen, Xinggang Wang, et al. Tinyclip: Clip distillation via affinity mimicking and weight inheritance. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 21970–21980, 2023. 7
- [38] Long Xing, Qidong Huang, Xiaoyi Dong, Jiajie Lu, Pan Zhang, Yuhang Zang, Yuhang Cao, Conghui He, Jiaqi Wang, Feng Wu, and Dahua Lin. Pyramidrop: Accelerating your large vision-language models via pyramid visual redundancy reduction, 2025. 1
- [39] Weihao Yu, Zhengyuan Yang, Linjie Li, Jianfeng Wang, Kevin Lin, Zicheng Liu, Xinchao Wang, and Lijuan Wang. Mm-vet: Evaluating large multimodal models for integrated capabilities. *ArXiv*, abs/2308.02490, 2023. 6
- [40] Xiang Yue, Yuansheng Ni, Kai Zhang, Tianyu Zheng, Ruoqi Liu, Ge Zhang, Samuel Stevens, Dongfu Jiang, Weiming Ren, Yuxuan Sun, et al. Mmmu: A massive multi-discipline multimodal understanding and reasoning benchmark for expert agi. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9556–9567, 2024. 6
- [41] Aohan Zeng, Xiao Liu, Zhengxiao Du, Zihan Wang, Hanyu Lai, Ming Ding, Zhuoyi Yang, Yifan Xu, Wendi Zheng, Xiao Xia, et al. Glm-130b: An open bilingual pre-trained model. *arXiv preprint arXiv:2210.02414*, 2022. 2
- [42] Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11975–11986, 2023. 1, 6
- [43] Peiyuan Zhang, Guangtao Zeng, Tianduo Wang, and Wei Lu. Tynllama: An open-source small language model. *arXiv preprint arXiv:2401.02385*, 2024. 2
- [44] Renshan Zhang, Yibo Lyu, Rui Shao, Gongwei Chen, Weili Guan, and Liqiang Nie. Token-level correlation-guided compression for efficient multimodal document understanding, 2024. 1
- [45] Baichuan Zhou, Ying Hu, Xi Weng, Junlong Jia, Jie Luo, Xien Liu, Ji Wu, and Lei Huang. Tynllava: A framework of small-scale large multimodal models. *arXiv preprint arXiv:2402.14289*, 2024. 1, 2, 6
- [46] Deyao Zhu, Jun Chen, Xiaoqian Shen, Xiang Li, and Mohamed Elhoseiny. Minigpt-4: Enhancing vision-language understanding with advanced large language models. *arXiv preprint arXiv:2304.10592*, 2023. 1, 2
- [47] Yuke Zhu, Chi Xie, Shuang Liang, Bo Zheng, and Sheng Guo. Focusllava: A coarse-to-fine approach for efficient and effective visual token compression, 2024. 1
- [48] Yichen Zhu, Minjie Zhu, Ning Liu, Zhicai Ou, Xiaofeng Mou, and Jian Tang. Llava-phi: Efficient multi-modal assistant with small language model, 2024. 1