To Preserve or To Compress: An In-Depth Study of Connector Selection in Multimodal Large Language Models

Junyan Lin[1], Haoran Chen[1], Dawei Zhu[2], Xiaoyu Shen[1]*
[1]Digital Twin Institute, Eastern Institute of Technology, Ningbo, China,
[2]Saarland University, Saarland Informatics Campus
linjyan00@gmail.com xyshen@eitech.edu.cn

Abstract

In recent years, multimodal large language models (MLLMs) have garnered significant attention from both industry and academia. However, there is still considerable debate on constructing MLLM architectures, particularly regarding the selection of appropriate connectors for perception tasks of varying granularities. This paper systematically investigates the impact of connectors on MLLM performance. Specifically, we classify connectors into feature-preserving and feature-compressing types. Utilizing a unified classification standard, we categorize sub-tasks from three comprehensive benchmarks, MMBench, MME, and SEED-Bench, into three task types: coarse-grained perception, fine-grained perception, and reasoning, and evaluate the performance. Our findings reveal that feature-preserving connectors excel in *fine-grained perception* tasks due to their ability to retain detailed visual information. In contrast, feature-compressing connectors, while less effective in fine-grained perception tasks, offer significant speed advantages and perform comparably in *coarse-grained perception* and *reasoning* tasks. These insights are crucial for guiding MLLM architecture design and advancing the optimization of MLLM architectures.

1 Introduction

Large language models (LLMs) have made significant advances in recent years, demonstrating remarkable capabilities in understanding and generating text (Brown et al., 2020; Su et al., 2022; Jiang et al., 2023; Bai et al., 2023; Touvron et al., 2023; Zhang et al., 2024; Su et al., 2024). Recently multimodal large language models (MLLMs) have emerged as a hot topic in both academia and industry due to their potential to handle multiple modalities, such as text and vision, in a unified framework (Wang et al., 2023; Alayrac et al., 2022; Gao et al., 2024). However, training a unified architecture from scratch across different modalities is often resource-intensive and time-consuming, making it feasible only for a limited number of large companies with substantial computational resources (Team, 2024; Zhou et al., 2024). As a result, researchers commonly adopt a **connector**-based approach, which fully leverages the existing powerful capabilities of a pre-trained language model (Li et al., 2023b; Liu et al., 2024b). This connector bridges the gap by transforming visual information from the encoder into vector representations that the LLM can process and understand. Through this method, the pre-trained text-based LLM is empowered to perceive and interpret visual data, enabling it to perform a wide range of visual tasks without requiring a complete retraining of the model from scratch (Yin et al., 2023).

Designing an optimal MLLM architecture remains a crucial and intriguing research area. While prior studies (Karamcheti et al., 2024; McKinzie et al., 2024; Laurençcon et al., 2024) have investigated various factors affecting MLLM performance, a significant gap persists in the detailed examination of the key component: **the connector**.

We categorize connectors into two types: **feature-preserving connectors**, which retain visual feature details by maintaining patch numbers, and **feature-compressing connectors**, which reduce computational load by abstracting patches into a specified number. Different studies have conflicting views: Lin et al. (2023); Chen et al. (2024) contends that feature-compressing connectors are suitable for coarse-grained perception tasks but perform weakly on fine-grained perception tasks. In contrast, McKinzie et al. (2024) observes little difference between the two. Although these studies provide experimental evidence, further exploration is needed to understand how connectors perform across varying perception granularities.

To address this gap, this paper aims to meticu-

*Corresponding Author

Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, pages 5666–5680
November 12-16, 2024

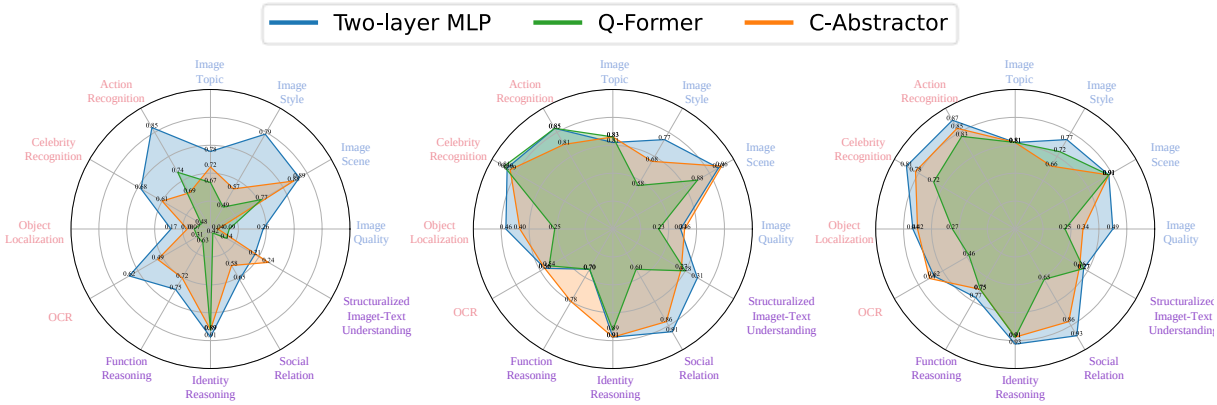


Figure 1: Comparison of radar chart performance at 224, 336, and 448 resolutions across coarse-grained perception, fine-grained perception, and reasoning tasks on MMBench. Each task includes four sub-tasks: Image Quality, Image Scene, Image Style, and Image Topic for coarse-grained perception; Action Recognition, Celebrity Recognition, Object Localization, and OCR for fine-grained perception; and Function Reasoning, Identity Reasoning, Social Relation, and Structuralized Image-Text Understanding for reasoning tasks.

lously investigate the effects of various connectors on tasks of different perception granularities. Building on the construction guidelines of MMBench (Liu et al., 2023), we evaluate the impact of connectors on the performance of MLLMs across three task types: coarse-grained perception, fine-grained perception, and reasoning. Our extensive experiments thoroughly explore connector performance across these varying perception granularities. Several noteworthy conclusions are drawn from testing on three multitask benchmarks. Figure 1 shows the performance of different connectors on the three tasks as well as the corresponding sub-tasks. Our main contributions are summarized as follows:

1. We conduct a comprehensive analysis of different connectors from multiple perspectives, including loss curves, compressed token number, image resolution, and performance metrics across tasks of different granularities.

2. We demonstrate that feature-compressing connectors significantly underperform in fine-grained perception tasks compared to feature-preserving connectors, while maintaining comparable performance in coarse-grained perception tasks.

3. We analyze the impact of different pooling methods within feature-compressing connectors, revealing that simpler pooling methods generally lead to more effective training and better overall performance.

2 Related Work

Connectors play a crucial role in aligning multimodal data within MLLMs, with various types. Based on whether the patch number of visual features is retained or reduced, we classify connectors into two categories: feature-preserving connectors and feature-compressing connectors.LLaVA (Liu et al., 2024b) employs a single-layer linear projection as its connector, whereas LLaVA-1.5 (Liu et al., 2024a) enhances this design by adding a GELU activation function and an extra linear projection. These feature-preserving connectors are designed to retain the details of visual features. Emu2 (Sun et al., 2024) uses a local average pooling strategy to standardize visual features into a uniform number of patches. BLIP-2 (Li et al., 2023b) utilizes the Q-Former, a cross-attention connector that uses a fixed number of learnable queries to interact with visual features, enabling global weighted pooling. HoneyBee (Cha et al., 2024) introduces the C-Abstractor, which utilizes convolutional neural networks to perform local weighted pooling based on Emu2's local average pooling, effectively extracting local features. These feature-compressing connectors adjust the length of feature patch number, thereby optimizing computational resources while reserving key information.

To explore the impact of various components and parameters in MLLMs on performance, numerous studies have been conducted. Karamcheti et al. (2024) rigorously investigates MLLMs along key design axes, such as optimization procedures, im-

age processing, pretrained visual representations, language models, and scaling properties. MM1 (McKinzie et al., 2024) aim to identify important design principles and lessons for constructing MLLMs through comprehensive ablation studies on architecture components, data choices, and training procedures. Additionally, Idefics2 (Laurençon et al., 2024) conducts extensive experiments around pre-trained models, architecture choice, data, and training methods to bring experimental clarity to core design choices in building MLLMs.

Their findings offer useful initial insights but require a more detailed analysis of task-specific performance for greater depth and applicability. Specifically, they lack a comprehensive examination of tasks with varying granularities, such as coarse-grained perception, fine-grained perception, and reasoning (Liu et al., 2023).

To address this, our paper thoroughly investigates the effects of various connectors on MLLMs, focusing on their performance across the aforementioned tasks. This comprehensive analysis aims to deepen our understanding of connectors' impact and guide targeted connector selection in future model design stages based on specific tasks.

3 Taxonomy of Connectors

3.1 Preliminaries

Multimodal large language models (MLLMs) generally consist of three key components: a visual encoder E, a connector C, and an LLM. For a given visual input V, the encoder E extracts visual features $f \in \mathbb{R}^{P \times d_v}$, where P is the number of visual patches and d_v is the channel dimension. The connector C, which is a crucial component, then aligns these visual features with the word embedding space as follows:

$$\begin{aligned} C : \mathbb{R}^{P \times d_v} &\rightarrow \mathbb{R}^{Q \times D} \\ x &= C(f) \end{aligned} \tag{1}$$

Here, $x \in \mathbb{R}^{Q \times D}$ represents the visual tokens that are input into the LLM, where Q is the number of visual tokens and D is the hidden size of the LLM. We categorize connectors into two types: feature-preserving connectors, where $P = Q$, and feature-compressing connectors, where $P > Q$.

3.2 Feature-Preserving Connector

Feature-preserving connectors maintain the patch number of visual features (i.e., $P = Q$) and are typically composed of components such as linear layers and activation layers. While they can retain detailed information, the computational complexity of the model grows exponentially with the visual token number. Existing feature-preserving connectors can be classified into linear and nonlinear types based on whether they include nonlinear operations. For example, the connector in LLaVA (Liu et al., 2024b) can be classified into linear type because it only contains a linear layer, as shown below:

$$x = Wf \tag{2}$$

where $W \in \mathbb{R}^{d_v \times D}$ is a trainable projection matrix, that maps the visual features $f \in \mathbb{R}^{P \times d_v}$ to the word embedding space. The connector used in LLaVA-1.5 (Liu et al., 2024a) can be classified into the nonlinear type because it incorporates an activation function and an additional linear layer on top of the basic linear type, as shown below:

$$x = W^{(2)}\phi(W^{(1)}f) \tag{3}$$

where $W^{(1)} \in \mathbb{R}^{d_v \times D}$ and $W^{(2)} \in \mathbb{R}^{D \times D}$ are trainable projection matrices, and ϕ denotes a nonlinear activation function GELU. The inclusion of the activation function allows the nonlinear connectors to capture more intricate patterns and dependencies in the data, enhancing the model's ability to manage complex visual-textual relationships.

3.3 Feature-Compressing Connector

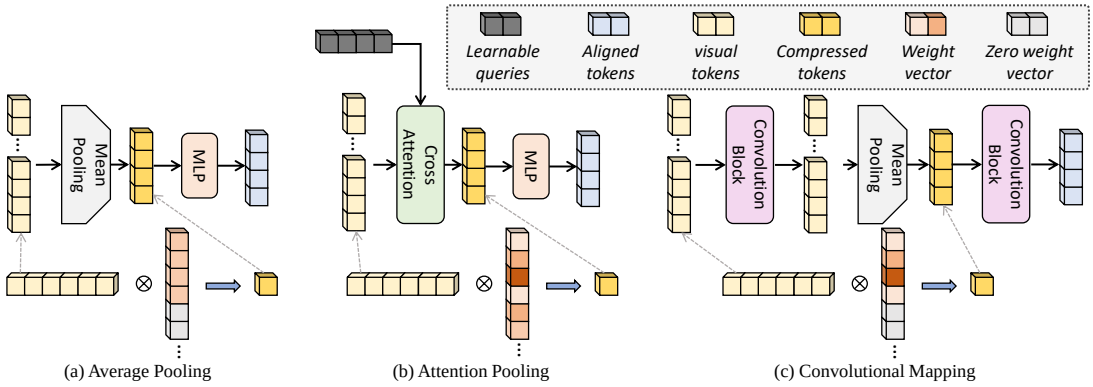
Feature-compressing connectors reduce the number of visual tokens Q through various strategies, aiming to preserve visual information while optimizing computational efficiency. Based on MM1 (McKinzie et al., 2024), we categorize feature-compressing connectors into three types: average pooling, attention pooling, and convolutional mapping, as shown in Figure 2. They generally operate in two steps. The first step involves using a pooling operation $\mathcal{P}$ to reduce the patch number P of visual feature f to Q ($Q < P$) as follows:

$$f' = \mathcal{P}(f) \tag{4}$$

where $f' \in \mathbb{R}^{Q \times d_v}$ represents the compressed visual features. The second step is consistent with the feature-preserving connector, where the compressed visual features f'_v are projected into the word embedding space using a transformation $\mathcal{T}$:

$$x = \mathcal{T}(f') \tag{5}$$

where $\mathcal{T}$ can be a multi-layer perception (MLP) or convolutional layer that maps f' to $x \in \mathbb{R}^{Q \times D}$.



(a) Average Pooling (b) Attention Pooling (c) Convolutional Mapping

Figure 2: The structure of different visual-language connectors. The upper part of the figure shows the overall structure of various connectors, while the lower part provides a simplified visualization during compression. (a) The Average Pooling-based connector compresses features by averaging visual tokens within local windows (b) The Attention Pooling-based connector uses cross-attention between learnable queries and visual tokens to abstract visual tokens into a certain number of compressed tokens. Each compressed token is derived from all visual tokens with weighted contributions. (c) The Convolutional Mapping-based connector uses convolution operations to enhances local context modeling while reducing the number of tokens. Each compressed token is derived from the visual tokens within local windows with weighted contributions.

Average Pooling This type of connector uses average pooling as $\mathcal{P}$ to reduce the number of tokens. For a given set of n feature patches in f, the average pooling operation can be formulated as follows:

$$f'_i = \frac{1}{n}\sum_{j=1}^{n} f_{(i-1)n+j} \tag{6}$$

where $f'_{v,i}$ represents the i-th averaged feature patch in f' and $f_{(i-1)n+j}$ represents the j-th feature patch in the i-th group of f. After obtaining the compressed visual features f', we directly apply the connector from LLaVA-1.5 as the transformation $\mathcal{T}$ to project f' into the word embedding space.

Attention Pooling This type of connector uses cross-attention as $\mathcal{P}$ to reduce the number of tokens. The patch number P is compressed by performing cross-attention between a set of learnable queries $Q \in \mathbb{R}^{Q\times d_c}$ and the visual features f, resulting in $f' \in \mathbb{R}^{Q\times d_c}$, where d_c is the hidden size of the cross-attention. The cross-attention mechanism can be formulated as follows:

$$\begin{aligned} K &= W_k f, \quad V = W_v f \\ A &= \text{softmax}\left(\frac{QK^\top}{\sqrt{d_c}}\right) \\ f'_i &= \sum_{j=1}^{P} A_{ij} V_j \end{aligned} \tag{7}$$

where $K, V \in \mathbb{R}^{P\times d_c}$ are the key and value matrices obtained by projecting the visual features f using the projection matrices $W_k, W_v \in \mathbb{R}^{d_v\times d_c}$, respectively. $A \in \mathbb{R}^{Q\times P}$ is the attention weight matrix, and A_{ij} represents the attention weight between the i-th query and the j-th visual feature. The compressed visual feature f'_i is obtained by weighted summation of the value vectors V_j. After obtaining the compressed visual features f', the transformation $\mathcal{T}$ is consistent with the approach used in average pooling connector.

Convolutional Mapping This type of connector uses a combination of convolutional layers and average pooling as $\mathcal{P}$ to reduce the number of tokens. The patch number P is compressed by first applying convolutional layers followed by average pooling, resulting in $f' \in \mathbb{R}^{Q\times d_v}$, where d_v is the channel dimension of the visual features. The transformation $\mathcal{T}$ is then applied using additional convolutional layers to project the compressed visual features into the word embedding space. The overall process can be formulated as follows:

$$\begin{aligned} f'_i &= \frac{1}{n}\sum_{j=1}^{n}\left(W_j \cdot f_{(i-1)n+j}\right) \\ x_i &= \sum_{k=-K}^{K} W'_k \cdot f'_{i+k} \end{aligned} \tag{8}$$

where x_i represents the i-th projected visual token, obtained by applying convolutional layers represented by weights W'_k over the compressed features f'_{i+k}. For simplification, the initial convolutional

layers and average pooling are represented as a local weighted average, denoted by W_j.

Characteristics Average pooling is a simple and efficient feature-compressing connector that quickly reduces patch numbers without adding any parameters, making it easy to train. However, it may lead to the loss of local information, reducing its effectiveness in fine-grained perception tasks. Attention pooling, utilizing a global weighted mechanism, retains more global information and offers a higher theoretical performance limit. Despite this, it has higher computational complexity and the most additional parameters due to the learnable queries and projection matrices, making it the most challenging to train. Furthermore, it may struggle with fine-grained tasks because the attention mechanism can find it difficult to preserve local image information (Dosovitskiy et al., 2020; Park and Kim, 2022). Convolutional mapping effectively preserves local details and involves a moderate number of parameters, striking a balance between parameter efficiency and the ability to capture fine-grained details. However, it lacks the capability to intuitively capture global features. These characteristics are intuitive, but their actual effectiveness and trade-offs need to be empirically validated through extensive experiments across tasks with varying perception granularities.

4 Experimental Settings

In this section, we present the experimental settings employed in our study, including the criteria for perception granularity (4.1), the benchmarks utilized for evaluation (4.2), and the specific implementation details of our models (4.3). Each aspect is carefully detailed to ensure a comprehensive understanding of our experiment.

4.1 Perception Granularity

The partition criterion for coarse-grained and fine-grained perception vary across different benchmarks. For example, MMBench categorizes tasks like 'Image Style' and 'Image Scene' under coarse-grained perception, which focuses on the global attributes and overall context of the image, while tasks like 'Object Localization' fall under fine-grained perception, focusing on the local details and specific features within the image. MME also differentiates between coarse-grained perception and fine-grained perception tasks, but its criteria focus more on testing the knowledge resources of MLLM, rather than the perspective of spatial scope.

For instance, the task of 'Object Localization' in MME is considered a coarse-grained perception task and 'Scene Recognition' is classified as a fine-grained perception task. However, in MMBench, they will be divided into coarse-grained perception task and fine-grained perception task, respectively. Figure 3 further illustrates this discrepancy. The left image is selected from the 'Color' sub-task, which is categorized as a coarse-grained perception task in MME. However, it actually focuses on local image details, which would reclassify it as a fine-grained perception task in MMBench. Conversely, the right image is selected from the 'Scene' sub-task, which is categorized as a fine-grained perception task in MME, but it actually focuses on the overall context of the image, making it a coarse-grained perception task in MMBench.

Figure 3: Examples of conflicting partition criterion for perception granularity in the MME benchmark.

Many methods exhibit conflicting views on the ability of different connectors to handle different granularities (Lin et al., 2023; Chen et al., 2024; McKinzie et al., 2024). To explore this issue, based on MMBench, we define coarse-grained perception as the ability to perceive image-level features, such as overall concepts and context. In contrast, fine-grained perception refers to object-level details within the image, such as identifying specific

features of individual objects. By analyzing performance in coarse-grained and fine-grained perception tasks, we can determine whether feature-preserving connectors and feature-compressing connectors excel in specific perception tasks. Additionally, by examining reasoning tasks, we can more accurately assess the impact of different types of connectors on the model's ability to understand and integrate multimodal information.

4.2 Benchmarks

To explore and evaluate various types of connectors, we utilize three well-established benchmarks with sub-task labels: MMBench (Liu et al., 2023), MME (Fu et al., 2023), and SEED-Bench (Li et al., 2023a). We reference the coarse-grained and fine-grained perception tasks defined above to reclassify the sub-tasks of MME and SEED-Bench. Detailed information on the sub-tasks reclassification can be found in Table 4, Table 5, and Table 6 in Appendix A.

4.3 Implementation Details

Given the focus of this paper on comparing connectors, we largely adhere to the configuration of LLaVA-1.5, with exceptions for connector modifications and using LLaMA 2 (Touvron et al., 2023) as the LLM. The visual encoder utilized is CLIP ViT-L/14 (Radford et al., 2021) with resolutions of 224 and 336. We keep the learning rate, batch size, training phases, and data usage consistent with LLaVA-1.5. For images with a resolution of 448, we refer to MM1 and employ position embedding interpolation to adapt CLIP ViT-L/14 from a resolution of 336 to 448. Considering that LoRA-based LLaVA-1.5 performs on par with the fully fine-tuning setting across the three benchmarks, we opt for the LoRA approach (Hu et al., 2021) to save computational resources. Refer to Table 7 in Appendix B for detailed connector configurations.

5 Results

To verify the ability of different connectors to perceive different image granularities and assess reasoning capabilities, we evaluate the performance of feature-preserving and feature-compressing connectors with different image resolutions across three key tasks: coarse-grained perception, fine-grained perception, and reasoning.

5.1 Effects of Feature-Preserving Connector

To assess the performance of feature-preserving connectors, we compare the linear type (referred to as the linear connector) with the nonlinear type (referred to as the two-layer MLP connector) across three tasks: coarse-grained perception, fine-grained perception, and reasoning. As shown in Figure 4, using two-layer MLP connector consistently outperform the linear connector in all task groups at a resolution below 448.

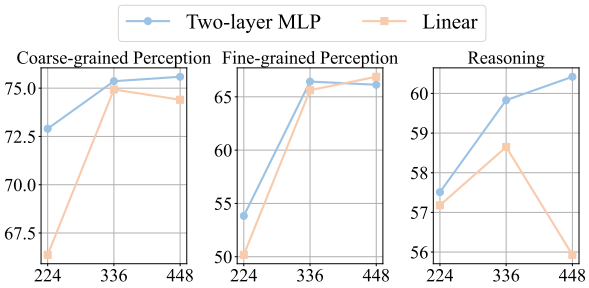


Figure 4: Comparison of two-layer MLP and linear connectors on coarse-grained, fine-grained perception, and reasoning tasks at resolutions of 224, 336, and 448.

When the resolution is increased to 448, although the linear connector performs on par with the two-layer MLP connector in fine-grained perception, it suffers a substantial performance drop in other tasks, particularly in reasoning. We hypothesize that a linear mapping may struggle balancing both perception and reasoning at high resolution. In contrast, the reasoning ability is further enhanced when using two-layer MLP with higher resolution.

5.2 Impact of Compressed Token Number

The compressed token number Q is an important parameter. We compare two widely used values: 64 and 144. We fix the resolution at 336 and evaluate average pooling, Q-Former, and C-Abstractor across three tasks. The results are shown in Figure 5. It can be seen that while 144 tokens generally provide a slight improvement in performance over 64 tokens, the difference is not substantial, indicating that both 64 and 144 tokens are adequate for robust image information extraction.

Additionally, to further demonstrate the impact of the compressed token number, we present the loss curves of different feature-compressing connectors with different compressed token numbers during the pretraining and finetuning stages in Figure 6 in Appendix C.1. It can be observed that the loss curves from 144 tokens show marginally better convergence than those from 64 tokens, especially

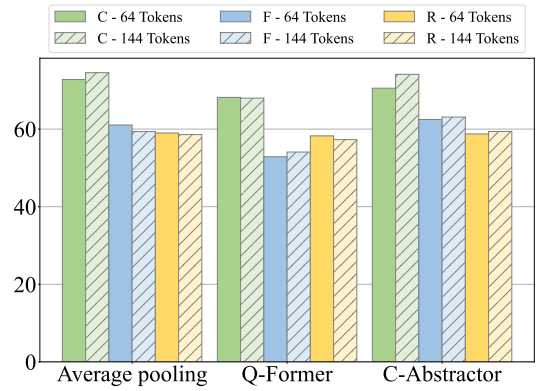


Figure 5: Analysis of the impact of different compressed token numbers on the performance of coarse-grained perception (C), fine-grained perception (F), and reasoning (R) tasks.

for C-Abstractor. Considering this slight performance gain, we ultimately choose 144 tokens as the standard setting for feature-compressing connectors for subsequent comparisons.

5.3 Impact of Image Resolution

We further explore the effect of image resolution on the metric performance of feature-preserving and feature-compressing connectors. Detailed experiments and analyses, as shown in Table 1, reveal that increasing the resolution from 224 to 336 enhances performance across all connector types for the three tasks, with the most significant improvements observed in fine-grained perception tasks, followed by coarse-grained perception tasks, and the least improvement in reasoning tasks. However, further increasing the resolution from 336 to 448 yields only marginal performance gains. Specifically, for feature-preserving connectors, the resolution increase from 224 to 336 results in improvements of 12.6% in fine-grained perception, 2.5% in coarse-grained perception, and 2.3% in reasoning tasks. For feature-compressing connectors, the improvements are 13.9%, 9.2%, and 4.3%, respectively. When the resolution is increased from 336 to 448, the performance changes for the former are 2.5%, 0.2%, and 0.6%, while for the latter, the changes are -0.5%, -1.0%, and 0.9%. We attribute this to the diminishing returns of higher resolutions and the current insufficient training data to support them.

Figure 6 in Appendix C.1 clearly illustrates the loss curves of all connectors at different resolutions. It can be seen that in most cases, increasing the resolution from 224 $\times$ 224 to 336 $\times$ 336 generally results in a decrease in training loss. However, when the resolution is further increased from 336 $\times$ 336 to 448 $\times$ 448, only the fine-tuning loss of the two-layer MLP decreases, while the others either remain unchanged or increase. This observation is consistent with the evaluation metrics.

Connectors	C	F	R
Two-layer MLP			
224	72.9	53.83	57.51
336	75.36	66.43	59.83
448	75.59	66.13	60.42
Average pooling			
224	70.11	53.47	57.56
336	74.54	59.37	58.59
448	74.29	64.84	60.53
Q-Former			
224	59.29	42.71	51.97
336	67.99	54.09	57.29
448	69.62	52.83	56.99
C-Abstractor			
224	64.93	49.33	55.05
336	74.12	63.11	59.39
448	73.05	62.62	60.31

Table 1: Comparison of two-layer MLP, average pooling with 144 tokens, C-Abstractor with 144 tokens, and Q-Former with 144 tokens on coarse-grained perception (C), fine-grained perception (F), and reasoning (R) tasks.

Connectors	C	F	R
Average pooling	74.29	64.84	60.53
Q-Former	69.62	52.83	56.99
C-Abstractor	73.05	62.62	60.31

Table 2: Comparison of feature-compressing connectors at 448 resolution on coarse-grained perception (C), fine-grained perception (F), and reasoning (R) tasks, each with the same compressed token number 144.

5.4 Effects of Feature-Compressing Connector

To explore the impact of different feature-compressing connectors on model performance, we conduct a detailed comparison on the three tasks under the settings of 448$\times$448 resolution and 144 compressed token number, as shown in Table 2. Overall, the performance of average pooling and C-Abstractor is similar, while Q-Former performs significantly worse. Specifically, for coarse-grained perception tasks, Q-Former does not show as large a performance gap compare to other connectors as it does in fine-grained perception tasks. This might be because, the self-attention mechanism disrupts the original positional information which is impor-

tant in fine-grained perception tasks. However, this does not fully explain why Q-Former also performs poorly in coarse-grained perception tasks. To explain this phenomenon, we present the training loss curves at Figure 6 in Appendix C.1. The curves show that Q-Former's loss decreases more slowly than the losses from other connectors. This indicates that Q-Former is more challenging to train, likely due to insufficient training data to support such a complex mechanism.

In summary, simple average pooling suffices for most tasks as LLMs can implicitly extract image information from visual tokens. Extensive interference in token extraction at the projector stage is not necessary. Complex connectors like Q-Former may require more aligned data for better results.

6 Suggestions for Connector Selection

In Section 5, we extensively discuss the performance of different connectors across various tasks. To consider efficiency and effectiveness simultaneously during the connector selection phase, we present the training times for models under different connectors in Table 3. It was observed that with increasing image resolution, the training costs for feature-compressing connectors change only slightly, whereas those for feature-preserving connectors significantly increase.

Based on the above evidences, we offer several recommendations for choosing connectors:

1. At an image resolution of 224, using a two-layer MLP is advisable as it significantly outperforms other connectors across the three tasks while maintaining a acceptable computational resource demand.

2. At an image resolution of 336, if the focus is on coarse-grained perception and reasoning tasks, the C-Abstractor and average pooling are recommended for their balance between efficiency and effectiveness. If fine-grained perception tasks are a priority, the two-layer MLP may be more suitable.

3. At an image resolution of 448, the token count for the two-layer MLP reaches 1024, which leads to excessive consumption of computational resources. Under these circumstances, C-Abstractor and average pooling 144tks emerge as more optimal choices. Specifically, the C-Abstractor reduces the training time by 80% in the pre-training stage and 51% in the fine-tuning stage compared to the two-layer MLP. This drastic reduction in training time not only makes the C-Abstractor and average pooling connectors more efficient but also significantly lowers the computational cost, making them highly suitable for scenarios with limited resources. The substantial decrease in training time at this high resolution highlights the importance of choosing the right connector to balance performance and resource usage.

These guidelines aim to assist in selecting the most appropriate connector, aligning with specific task requirements and resource availability.

Resolution	Connectors	Tokens	Stage	Time (hours)
224	Two-layer MLP	256	1	2.0
		256	2	11.3
	C-Abstractor	144	1	0.8 (↓60%)
		144	2	8.0 (↓29%)
336	Two-layer MLP	576	1	3.6
		576	2	11.9
	C-Abstractor	144	1	1.2 (↓67%)
		144	2	8.0 (↓33%)
448	Two-layer MLP	1024	1	6.5
		1024	2	16.5
	C-Abstractor	144	1	1.3 (↓80%)
		144	2	8.1 (↓51%)

Table 3: Training time for different connectors. Stage 1 refers to the pre-training stage, and Stage 2 refers to the fine-tuning stage. All training is conducted in an environment with 8 A800 GPUs.

7 Conclusion

In this paper, we conduct extensive experiments to evaluate commonly used connectors in MLLMs. Our findings indicate that although feature-preserving connectors generally offer the best performance, their advantage over feature-compressing connectors diminishes as resolution increases, while their computational costs rise exponentially. Among feature-compressing connectors, average pooling and C-Abstractor outperform Q-Former, consistently delivering better results across all resolutions and task granularities. Our results clearly demonstrate that the choice of connector depends on the resolution, task granularity, and computational budget. Based on these findings, we offer guidance on selecting connectors to balance both effectiveness and efficiency.

Limitations

In alignment with the base configuration of LLaVA-1.5, our approach involves using positional encod-

ing interpolation to scale images from 336x336 to 448x448, rather than employing a visual encoder that natively supports the 448x448 resolution. This method may lead to suboptimal results. Another limitation is that our training data also comes from LLaVA-1.5, resulting in a relatively small total number of training samples, only 1.23 M. In contrast, the InstructBLIP (Dai et al., 2024) with use Q-Former as connector has 130.2 M training samples. This significant difference in the number of training samples might render our conclusions inapplicable in scenarios with a large volume of training data. In the future, we could explore this discrepancy using a larger set of training samples.

Acknowledgement

We sincerely thank the reviewers of this work for their constructive and insightful feedback. We thank Xingluan (AI Cloud computing service), EIT and IDT High Performance Computing Center for providing computational resources for this project.

References

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*, 35:23716–23736.

Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. 2023. Qwen-vl: A versatile vision-language model for understanding, localization, text reading, and beyond. *arXiv preprint arXiv:2306.01595*.

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*, 33:1877–1901.

Junbum Cha, Wooyoung Kang, Jonghwan Mun, and Byungseok Roh. 2024. Honeybee: Locality-enhanced projector for multimodal llm. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13817–13827.

Gongwei Chen, Leyang Shen, Rui Shao, Xiang Deng, and Liqiang Nie. 2024. Lion: Empowering multimodal large language model with dual-level visual knowledge. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 26540–26550.

Wenliang Dai, Junnan Li, Dongxu Li, Anthony Meng Huat Tiong, Junqi Zhao, Weisheng Wang, Boyang Li, Pascale N Fung, and Steven Hoi. 2024. Instructblip: Towards general-purpose vision-language models with instruction tuning. *Advances in Neural Information Processing Systems*, 36.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

Chaoyou Fu, Peixian Chen, Yunhang Shen, Yulei Qin, Mengdan Zhang, Xu Lin, Zhenyu Qiu, Wei Lin, Jinrui Yang, and Xiawu Zheng. 2023. Mme: A comprehensive evaluation benchmark for multimodal large language models. In *arXiv preprint arXiv:2306.13394*.

Peng Gao, Renrui Zhang, Chris Liu, Longtian Qiu, Siyuan Huang, Weifeng Lin, Shitian Zhao, Shijie Geng, Ziyi Lin, Peng Jin, et al. 2024. Sphinx-x: Scaling data and parameters for a family of multi-modal large language models. *arXiv preprint arXiv:2402.05935*.

Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. 2017. 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.

Danna Gurari, Qing Li, Abigale J Stangl, Anhong Guo, Chi Lin, Kristen Grauman, Jiebo Luo, and Jeffrey P Bigham. 2018. Vizwiz grand challenge: Answering visual questions from blind people. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3608–3617.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

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*, pages 6700–6709.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Siddharth Karamcheti, Suraj Nair, Ashwin Balakrishna, Percy Liang, Thomas Kollar, and Dorsa Sadigh. 2024. Prismatic vlms: Investigating the design space of visually-conditioned language models. *arXiv preprint arXiv:2402.07865*.

Sahar Kazemzadeh, Vicente Ordonez, Mark Matten, and Tamara Berg. 2014. Referitgame: Referring to objects in photographs of natural scenes. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 787–798.

Hugo Laurençcon, Léo Tronchon, Matthieu Cord, and Victor Sanh. 2024. What matters when building vision-language models? *arXiv preprint arXiv:2405.02246*.

Bohao Li, Rui Wang, Guangzhi Wang, Yuying Ge, Yixiao Ge, and Ying Shan. 2023a. Seed-bench: Benchmarking multimodal llms with generative comprehension. In *arXiv preprint arXiv:2307.16125*.

Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. 2023b. 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.

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

Ziyi Lin, Chris Liu, Renrui Zhang, Peng Gao, Longtian Qiu, Han Xiao, Han Qiu, Chen Lin, Wenqi Shao, Keqin Chen, et al. 2023. Sphinx: The joint mixing of weights, tasks, and visual embeddings for multi-modal large language models. *arXiv preprint arXiv:2311.07575*.

Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. 2024a. Improved baselines with visual instruction tuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 26296–26306.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2024b. Visual instruction tuning. *Advances in Neural Information Processing Systems*, 36.

Yuan Liu, Haodong Duan, Yuanhan Zhang, Bo Li, Songyang Zhang, Wangbo Zhao, Yike Yuan, Jiaqi Wang, Conghui He, Ziwei Liu, et al. 2023. Mmbench: Is your multi-modal model an all-around player? *arXiv preprint arXiv:2307.06281*.

Pan Lu, Swaroop Mishra, Tanglin Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. 2022. Learn to explain: Multimodal reasoning via thought chains for science question answering. *Advances in Neural Information Processing Systems*, 35:2507–2521.

Brandon McKinzie, Zhe Gan, Jean-Philippe Fauconnier, Sam Dodge, Bowen Zhang, Philipp Dufter, Dhruti Shah, Xianzhi Du, Futang Peng, Floris Weers, et al. 2024. Mm1: Methods, analysis & insights from multimodal llm pre-training. *arXiv preprint arXiv:2403.09611*.

Namuk Park and Songkuk Kim. 2022. How do vision transformers work? *arXiv preprint arXiv:2202.06709*.

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*, pages 8748–8763. PMLR.

Amanpreet Singh, Vivek Natarajan, Meet Shah, Yu Jiang, Xinlei Chen, Dhruv Batra, Devi Parikh, and Marcus Rohrbach. 2019. Towards vqa models that can read. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8317–8326.

Hui Su, Zhi Tian, Xiaoyu Shen, and Xunliang Cai. 2024. Unraveling the mystery of scaling laws: Part i. *arXiv preprint arXiv:2403.06563*.

Hui Su, Xiao Zhou, Houjin Yu, Xiaoyu Shen, Yuwen Chen, Zilin Zhu, Yang Yu, and Jie Zhou. 2022. Welm: A well-read pre-trained language model for chinese. *arXiv preprint arXiv:2209.10372*.

Quan Sun, Yufeng Cui, Xiaosong Zhang, Fan Zhang, Qiying Yu, Yueze Wang, Yongming Rao, Jingjing Liu, Tiejun Huang, and Xinlong Wang. 2024. Generative multimodal models are in-context learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14398–14409.

Chameleon Team. 2024. Chameleon: Mixed-modal early-fusion foundation models. *arXiv preprint arXiv:2405.09818*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Weihan Wang, Qingsong Lv, Wenmeng Yu, Wenyi Hong, Ji Qi, Yan Wang, Junhui Ji, Zhuoyi Yang, Lei Zhao, Xixuan Song, et al. 2023. Cogvlm: Visual expert for pretrained language models. *arXiv preprint arXiv:2311.03079*.

Shukang Yin, Chaoyou Fu, Sirui Zhao, Ke Li, Xing Sun, Tong Xu, and Enhong Chen. 2023. A survey on multimodal large language models. *arXiv preprint arXiv:2306.13549*.

Miaoran Zhang, Vagrant Gautam, Mingyang Wang, Jesujoba O Alabi, Xiaoyu Shen, Dietrich Klakow, and Marius Mosbach. 2024. The impact of demonstrations on multilingual in-context learning: A multidimensional analysis. *arXiv preprint arXiv:2402.12976*.

Chunting Zhou, Lili Yu, Arun Babu, Kushal Tirumala, Michihiro Yasunaga, Leonid Shamis, Jacob Kahn, Xuezhe Ma, Luke Zettlemoyer, and Omer Levy. 2024. Transfusion: Predict the next token and diffuse images with one multi-modal model. *arXiv preprint arXiv:2408.11039*.

Appendix

We provide some additional information as supplementary material. This appendix is divided into three sections:

- Details of sub-task reclassification for MME versus SEED-Bench are presented in Appendix A;
- Experimental details are presented in Appendix B;
- Additional results are presented in Appendix C;

A Sub-Task Reclassification

The parent tasks for sub-tasks of SEED-Bench and MME before and after reclassification are shown in Table 4, Table 5 respectively. Table 6 shows all the sub-tasks of MMBench, MME, and SEED-Bench and their reclassified parent tasks.

B Experimental Details

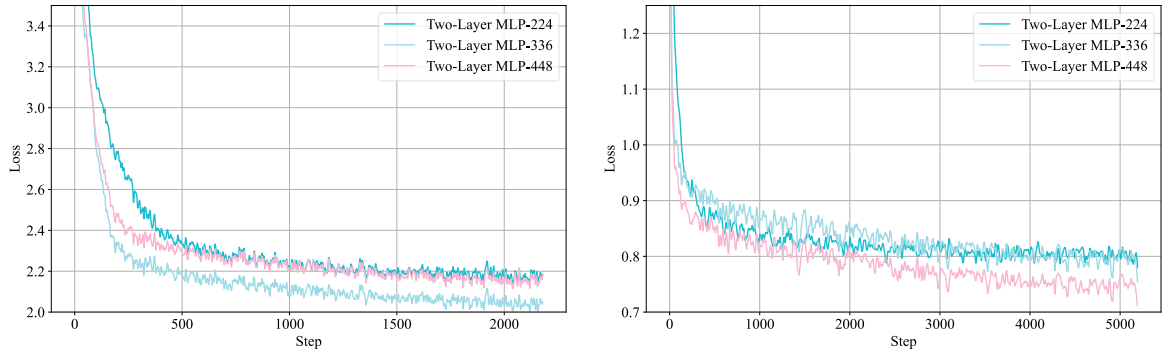
In this section, we further present the experimental settings, specifically the detailed list of training configurations for different connectors, as shown in Table 7.

C Additional Results

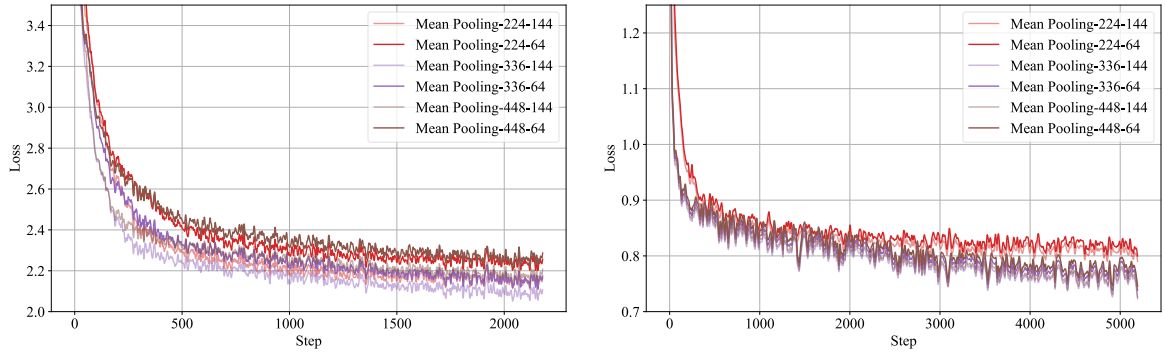
This section provides additional experimental results and analyses to supplement the findings presented in the main text. We include detailed evaluations of loss curves and comprehensive results from additional benchmarks to provide a more thorough understanding of the performance of different connectors.

C.1 Loss curves for different connectors

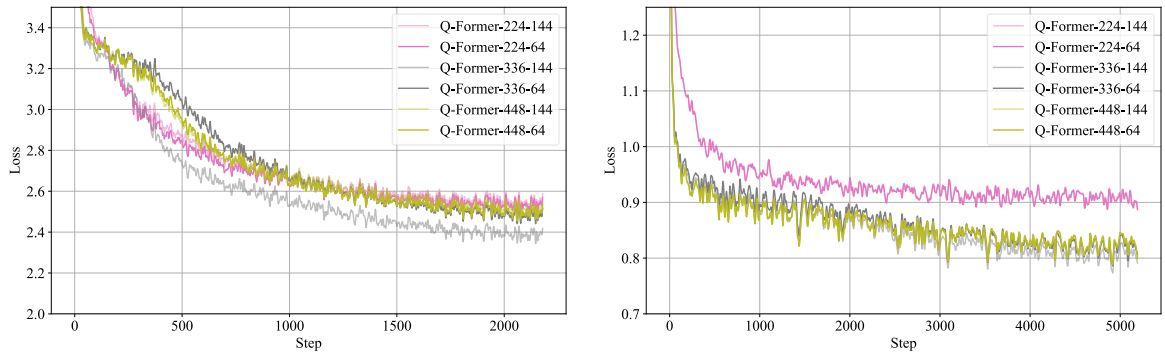
Loss curves of the model with different connectors are shown in Figure 6. The loss curves provide several insights that corroborate the findings in the main text: 1. For feature-compressing connectors, the difficulty of convergence increases with the number of parameters and complexity, following the trend: Q-Former > C-Abstractor > Average pooling. 2. When comparing different compressed token numbers, their convergence curves are very similar, with 144 tokens performing slightly better than 64 tokens. 3. Among the image resolutions of 224, 336, and 448, the resolution of 336 often shows significant improvement over 224, but the



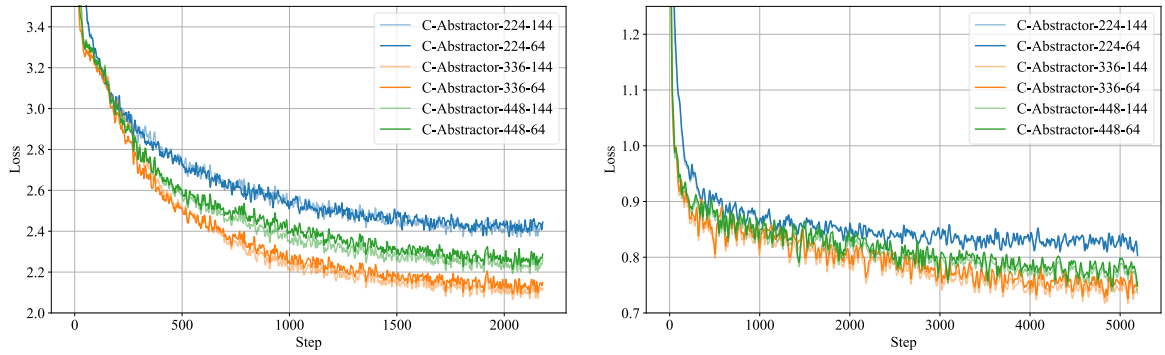
(a) The training loss curves of Two-Layer MLP-based model



(b) The training loss curves of Mean Pooling-based model



(c) The training loss curves of Q-Former-based model



(d) The training loss curves of C-Abstractor-based model

Figure 6: Loss curves of different connectors during the pretrain and finetune stages. The left plot shows the training loss during the pretrain stage, and the right plot shows the loss during the finetune stage. The legend in the upper right corner follows the format: connector class-image size-token number (e.g., C-Abstractor-224-144, where the connector is C-Abstractor, the image resolution is 224×224, and the number of tokens is 144). It can be observed that for feature-compressing connectors, the number of tokens has little effect on the loss, while image resolution significantly impacts the model.

Original Parent Task	Original Sub-Task	New Parent Task
Spatial Understanding	Scene Understanding	Coarse-grained Perception
	Instance Identity	Fine-grained Perception
	Instance Attribute	Fine-grained Perception
	Instance Location	Fine-grained Perception
	Instance Counting	Fine-grained Perception
	Spatial Relation	Fine-grained Perception
	Instance Interaction	Fine-grained Perception
	Visual Reasoning	Fine-grained Perception
	Text Recognition	Fine-grained Perception

Table 4: Sub-Task Reclassification for SEED-Bench.

Original Parent Task	Original Sub-Task	New Parent Task
Perception (Coarse-grained)	Existence	Fine-grained Perception
	Count	Fine-grained Perception
	Position	Fine-grained Perception
	Color	Fine-grained Perception
Perception (Fine-grained)	Poster	Coarse-grained Perception
	Celebrity	Fine-grained Perception
	Scene	Coarse-grained Perception
	Landmark	Coarse-grained Perception
	Artwork	Coarse-grained Perception
Perception (OCR)	OCR	Fine-grained Perception

Table 5: Sub-Task Reclassification for MME.

difference between 448 and 336 is not pronounced, with each resolution occasionally outperforming the other in different scenarios.

C.2 Evaluation results on more benchmarks

To achieve a more extensive and comprehensive comparison, aligning with other works, we conduct experiments on 9 additional benchmarks without sub-task information. These benchmarks include TextVQA (Singh et al., 2019), POPE (Li et al., 2023c), VQAv2 (Goyal et al., 2017), ScienceQA (Lu et al., 2022), GQA (Hudson and Manning, 2019), RefCOCO, RefCOCO+, RefCOCOg (Kazemzadeh et al., 2014), and VizWiz (Gurari et al., 2018). The results are shown in Table 8. The metrics used are Exact Match for TextVQA, F1-Score for POPE, Accuracy for VQAv2, GQA, ScienceQA, MMBench, and VizWiz, and CIder for RefCOCO. For SEED-Bench, Accuracy is calculated only on image data.

Tasks	From	Sub-Tasks
Coarse	MMBench	Image Quality
		Image Topic
		Image Emotion
		Image Scene
		Image Style
	MME	Artwork
		Landmark
		Posters
		Scene
	SEED-Bench	Scene Understanding
Fine-grained	MMBench	OCR
		Celebrity Recognition
		Object Localization
		Attribute Recognition
		Action Recognition
		Attribute Comparison
		Spatial Relationship
	MME	OCR
		Celebrity
		Color
		Count
		Existence
		Position
	SEED-Bench	Instance Identity
		Instance Attribute
		Instance Location
		Instance Counting
		Spatial Relationship
		Instance Interaction
Reasoning	MMBench	Function Reasoning
		Identity Reasoning
		Physical Property Reasoning
		Future Prediction
		Image-Text Understanding
		Nature Relation
		Physical Relation
		Social Relation
	MME	Code Reasoning
		Commonsense Reasoning
		Numerical Calculation
		Text Translation
	SEED-Bench	Visual Reasoning

Table 6: Sub-Tasks of MMBench, MME, and SEED-Bench and Their Reclassified Parent Tasks

Connector Type	Subclass	Resolution	Token Number
Feature-Preserving	Linear	224, 336, 448	-
	Nonlinear		-
Feature-Compressing	Average Pooling	224, 336, 448	64, 144
	Attention Pooling		64, 144
	Convolutional Mapping		64, 144

Table 7: Connector Configurations.

Resolution	Projectors	TextVQA	POPE	VQAv2	ScienceQA	GQA	MME^P	MMB	SEED	Refcoco	Refcoco+	Refcocog	VizWiz
244	Linear	46.88	81.06	69.54	63.97	50.29	1227.7	50.94	53.53	15.96	15.58	46.07	53.3
	Two-layers MLP	49.08	81.89	73.26	64.77	53.8	1309.91	55.67	58.3	23.2	23.03	52.79	55.26
	Average pooling 64tks	46.66	80.53	70.52	64.61	52.25	1294.79	53.78	56.3	20.09	19.43	49.14	55.63
	Average pooling 144tks	49.61	82.03	73.17	64.18	53.75	1333.86	54.04	58.63	23.44	23.32	54.57	54.7
	Q-Former 64tks	34.47	83.07	58.33	58.29	35.8	1050.59	36.77	46.09	8.82	8.34	32.22	24.04
	Q-Former 144tks	33.49	76.98	57.81	60.53	28.36	1127.66	37.2	45.8	2.66	2.49	13.41	18.06
	C-Abstactor 64tks	42.29	80.15	66.55	63.1	51.39	1228.72	48.79	52.74	18.45	17.74	50.65	53.9
	C-Abstactor 144tks	43.65	81.19	66.46	63.57	50.89	1204.11	48.45	52.87	9	8.75	38.11	52.87
336	Linear	56.16	86.06	78.41	70.08	62.53	1431.73	65.38	65.74	28.64	27.62	63.03	55.58
	Two-layers MLP	56.37	85.63	78.65	70.97	63.24	1486.42	65.12	67.08	28.65	28.15	62.82	56.22
	Average pooling 64tks	50.99	84.01	74.76	69.77	59.21	1388.58	63.05	61.49	24.73	23.77	57.97	51.84
	Average pooling 144tks	53.88	85.03	76.4	70.17	60.14	1457.42	63.4	63.98	28.88	28.71	60.3	53.54
	Q-Former 64tks	45.34	80.97	67.48	70.41	53.82	1244.26	57.73	53.73	25.5	24.81	57.14	49.88
	Q-Former 144tks	46.87	76.07	67.16	70.88	53.34	1240.74	58.59	53.09	29.6	27.99	57.28	49.52
	C-Abstactor 64tks	54.34	85.21	76.77	70.05	60.38	1360	63.49	62.55	25.35	24.51	61.22	54.37
	C-Abstactor 144tks	54.36	85.07	76.77	71.4	61.07	1450.46	63.57	63.42	28.76	28.6	60.71	55.13
448	Linear	55.73	84.92	77.65	68.56	60.93	1458	63.74	65.03	29.98	29.44	62.17	54.41
	Two-layers MLP	56.32	84.55	78.97	69.42	61.28	1516.15	65.12	66.74	28.75	28.66	62.56	57.93
	Average pooling 64tks	50.54	82.97	75.07	70.01	59.19	1422.03	62.29	62.19	27.98	27.89	62.01	56.76
	Average pooling 144tks	54.32	84.55	68.3	69.42	60.57	1474.24	63.48	64.91	27.83	27.58	64.05	53.28
	Q-Former 64tks	45.41	81.72	67.26	69.37	52.28	1281.34	56.96	54.49	27.4	26.51	55.07	46.98
	Q-Former 144tks	44.82	81.2	66.3	69.49	52.34	1236.01	57.82	52.86	26.39	25.71	56.07	50.88
	C-Abstactor 64tks	50.44	83.02	74.65	69.72	58.3	1374.8	62.71	61.67	25.99	25.7	57.41	54.62
	C-Abstactor 144tks	51.9	84.51	75.71	69.7	59.55	1433.33	63.23	62.49	27.98	27.75	59.85	55.3

Table 8: Performance of different connectors across 12 datasets at resolutions of 224, 336, and 448. Here, MME^P denotes the MME-Perception tasks, MMB stands for MMBench, and SEED indicates SEED-Bench.