Journal of Machine Learning Research 26 (2025) 1-61 Submitted 12/24; Revised 8/25; Published 10/25

Scaling Capability in Token Space: An Analysis of Large Vision Language Model

Tenghui Li TENGHUI.LEE@FOXMAIL.COM
School of Automation, Guangdong University of Technology, Guangzhou 510006, China

Guoxu Zhou * GX.ZHOU@GDUT.EDU.CN
School of Automation, Guangdong University of Technology, Guangzhou 510006, China
Key Laboratory of Intelligent Detection and the Internet of Things in Manufacturing, Ministry of Education, Guangzhou, China
Guangdong Provincial Key Laboratory of Intelligent Systems and Optimization Integration(GDUT), Guangzhou 510006, China

Xuyang Zhao XUYANG.ZHAO@RIKEN.JP
Medical Science Data-driven Mathematics Team, RIKEN Center for Interdisciplinary Theoretical and Mathematical Sciences, Yokohama 230-0045, Japan
Medical Data Mathematical Reasoning Special Team, RIKEN Center for Integrative Medical Sciences, Yokohama 230-0045, Japan
Department of Artificial Intelligence Medicine, Chiba University, Chiba 260-0856, Japan

Qibin Zhao QIBIN.ZHAO@RIKEN.JP
Tensor Learning Team, RIKEN Center for Advanced Intelligence Project, Tokyo 103-0027, Japan
School of Automation, Guangdong University of Technology, Guangzhou 510006, China

Editor: Kai-Wei Chang

Abstract

Large language models have demonstrated predictable scaling behaviors with respect to model parameters and training data. This study investigates whether a similar scaling relationship exist for vision-language models with respect to the number of vision tokens. A mathematical framework is developed to characterize a relationship between vision token number and the expected divergence of distance between vision-referencing sequences. The theoretical analysis reveals two distinct scaling regimes: sublinear scaling for less vision tokens and linear scaling for more vision tokens. This aligns with model performance relationships of the form $S(n) \approx c/n^{\alpha(n)}$, where the scaling exponent relates to the correlation structure between vision token representations. Empirical validations across multiple vision-language benchmarks show that model performance matches the prediction from scaling relationship. The findings contribute to understanding vision token scaling in transformers through a theoretical framework that complements empirical observations.

Keywords: large language models, vision large language models, token-efficiency scaling capability, vision tokens, theoretical analysis

---

*. corresponding author

©2025 Tenghui Li, Guoxu Zhou, Xuyang Zhao, and Qibin Zhao.

License: CC-BY 4.0, see `https://creativecommons.org/licenses/by/4.0/`. Attribution requirements are provided at `http://jmlr.org/papers/v26/24-2243.html`.

1. Introduction

Large language models (LLMs) have demonstrated remarkable capabilities across diverse natural language processing tasks (Brown et al., 2020; OpenAI, 2023; Anthropic, 2024; Touvron et al., 2023; Dubey et al., 2024; Bai et al., 2023a). The successes of these transformer-based architectures have motivated their extension to multimodal settings (Bavishi et al., 2023; Alayrac et al., 2022; Liu et al., 2023; Bai et al., 2023b; Chen et al., 2024b; Lu et al., 2024; Hu et al., 2024; Li et al., 2023b). This integration is typically achieved by employing vision encoders to convert images into discrete tokens that can be processed alongside text tokens within the unified transformer framework.

A fundamental challenge in multimodal transformers is the substantial number of vision tokens generated from images. For example, CLIP ViT-L/14 produces 256 tokens from a (224×224) image, while ViT-H/14 generates 576 tokens from a (336×336) image (Radford et al., 2021). High-resolution models like InternLM-XComposer2-4KHD can produce up to 2377 tokens for a 4K image (Dong et al., 2024), leading to significant computational costs. Conversely, efficiency-focused approaches like BLIP-2's Q-Former compress visual information into only 32 fixed tokens (Li et al., 2023b), raising questions about potential information loss. This yields a trade-off between computational efficiency and performance preservation, which motivates us to investigate the relationship between the number of vision tokens and model performance in vision language tasks. A fundamental question arises:

> *What is the relationship between the number of vision tokens and model performance in vision language tasks?*

To systematically investigate this fundamental question, there are some key aspects that need to be addressed. First, a principled methodology is required to analyze the relationship under more general conditions. Second, mathematical operations can be formulated to characterize the relationship. Third, it is also important to validate the findings in real world vision language benchmarks. These considerations naturally lead to three interconnected research questions:

1. Can a mathematical framework be formulated to characterize the relationship between vision tokens and model performance?
2. What are the theoretical properties of this relationship, and do scaling behaviors similar to those observed in language models also exist for vision tokens in vision-language models?
3. How can these findings be validated across diverse vision language benchmarks under real world conditions?

These three questions are addressed through three complementary approaches that build understanding from theory setup to evaluation under real-world large vision language benchmarks.

Mathematical framework. A mathematical framework is developed to characterize how the number of vision tokens affects model performance. The core approach is to measure the representational distance between a model's hidden states when processing two distinct input sequences that share identical prefix tokens but diverge in their vision-referencing components. The framework operates based on a fundamental insight: a model's ability to distinguish between similar inputs serves as a proxy for its discriminative capability and, consequently, its performance on vision-language tasks. When the representational distance between two branching sequences is small, the model struggles to differentiate between them, leading to ambiguous predictions and reduced performance. Conversely, when this distance is large, the model can reliably distinguish between the inputs, enabling confident and accurate responses.

Theoretical analysis and scaling properties. The analysis reveals that model performance follows predictable mathematical relationships with the number of vision tokens. The expected divergence of distance between different vision-referencing sequences $\mathcal{D}(n)$ scales as

$$\mathbb{E}_{\mathcal{V}}\left[\mathcal{D}(n)\right] = \mathcal{O}\left(\sqrt{n\left(1-\psi_{\text{equal}}^{(AB)}(n)\right)+(n^2-n)\left(\psi_{\text{cross}}^{(AA)}(n)-\psi_{\text{cross}}^{(AB)}(n)\right)}\right), \tag{1}$$

where n represents the number of differing vision tokens and the dependency measures $\psi_{\text{equal}}^{(AB)}(n)$, $\psi_{\text{cross}}^{(AA)}(n)$, and $\psi_{\text{cross}}^{(AB)}(n)$ capture the correlation structure between hidden representations. The divergence bound exhibits two distinct scaling regimes: sublinear scaling $\mathcal{O}\left(\sqrt{n}\right)$ for small n when the linear component dominates, and linear scaling $\mathcal{O}\left(n\right)$ for large n when the quadratic component dominates. This mathematical characterization aligns with scaling relationships analogizing to the scaling laws discovered in language models with respect to parameters and training data (Kaplan et al., 2020; Hoffmann et al., 2022). The scaling has the form of

$$S(n) \approx \frac{c}{n^{\alpha(n)}}, \tag{2}$$

where $S(n)$ denotes model performance, c is a scaling constant, and $\alpha(n)$ is the scaling exponent. Under specific conditions, the scaling exponent $\alpha(n)$ can be expressed as $\alpha(n) = \frac{\beta}{2}\log_n\left(\frac{1+n|_{\rho(n)=1}}{n+n|_{\rho(n)=1}}\right)-\frac{\beta}{2}$, where $n|_{\rho(n)=1}$ represents a critical balance point between $\mathcal{O}\left(\sqrt{n}\right)$ and $\mathcal{O}\left(n\right)$ scaling regimes, and β is a constant that relates model performance to the expected divergence. For a more comprehensive discussion of these scaling properties in token space, please refer to Subsection 3.4.

Model architecture and validation. A specific vision language model architecture is designed to evaluate the theoretical predictions in practice. Following the popular architecture design of vision language models (Liu et al., 2023; Guo et al., 2024; Dong et al., 2024), the model is built on vision encoder as vision token generator and large language model as the backbone. One main distinct requirement is that the architecture can be easily adapted to generate different numbers of vision tokens for controlled testing. Comprehensive experiments across multiple vision-language benchmarks are conducted to validate the theoretical predictions and demonstrate the effectiveness of the architecture. Results show that the model's performance follows the predicted scaling relationship, providing empirical evidence for the theoretical framework's accuracy.

Paper organization. The remainder of this paper is organized as follows. Relevant prior work is reviewed in Section 2. A theoretical framework for analyzing vision token scaling through sensitivity analysis is presented in Section 3. The proposed model architecture, designed to systematically investigate the scaling relationships, is described in Section 4. Comprehensive experimental validation across multiple benchmarks is provided in Section 5. Finally, conclusions, including implications and directions for future research, are drawn in Section 6.

2. Related Works

This section reviews relevant prior work across three key areas: vision language model architectures, scaling laws in neural models, and approaches to vision token optimization. The relationship between these areas and the present work is clarified to establish the distinctive contributions of this research.

Vision large language models. Vision language models integrate visual and textual information to enable vision capabilities in language models. Some methods incorporate specialized structures to facilitate cross-modal interactions. For example, Flamingo (Alayrac et al., 2022) applies cross-attention mechanisms between vision and language tokens through gated cross-attention layers, while CogVLM (Wang et al., 2024) incorporates vision expert modules comprising specialized attention and MLP components within each transformer layer. A more prevalent approach leverages existing language models by concatenating vision tokens alongside text tokens, which has become the dominant paradigm in recent vision language models such as LLaVA (Liu et al., 2023), MiniGPT-4 (Zhu et al., 2024) and other vision language models.

The number of vision tokens varies significantly across different methods and represents a critical design consideration. InternLM-XComposer2-4KHD (Dong et al., 2024) and LLaVA-UHD (Guo et al., 2024) utilize high-resolution vision processing and produce thousands of tokens per image, which is computationally expensive but enables detailed visual understanding. Conversely, other methods aim to reduce the number of vision tokens while maintaining performance. BLIP-2 (Li et al., 2023b) compresses visual information into 32 fixed tokens through its Q-Former architecture, demonstrating substantial token reduction with maintained performance. VisPrune (Zhang et al., 2024) and VisionZip (Yang et al., 2025) apply pruning and merging strategies to reduce token requirements, while VoCo-LLaMA (Ye et al., 2025) introduces a learnable compression token to reduce vision KV cache during inference.

Scaling properties. Scaling laws in large language models describe the performance of language models with respect to model size, training data, and computational resources. Kaplan scaling (Kaplan et al., 2020) established that language model performance follows predictable power-law relationships with model parameters N and dataset size D, where the cross-entropy loss L scales according to power-law relationships. Subsequent work Chinchilla scaling (Hoffmann et al., 2022) refined these relationships for optimal resource allocation, establishing improved scaling coefficients and training procedures. Research by Muennighoff et al. (2025) investigates scaling laws with limited amounts of unique data.

Scaling with the number of vision tokens. Recent research has begun investigating vision token utilization from practical optimization perspectives, though with different objectives and methodologies than the present work. Research Li et al. (2024c) investigate trade-offs between large models with fewer visual tokens and smaller models with more visual tokens under given computational budgets. Their work derives empirical scaling relationships through experiments across various model sizes and vision token configurations, focusing on inference-time optimization and resource allocation. The present research differs fundamentally from this prior work in several key aspects. First, the research scope addresses different fundamental questions: Li et al. (2024c) optimizes inference configurations by balancing model size against the number of vision tokens under computational budgets.

This research investigates the intrinsic mathematical relationship between vision token quantity and model discriminative capacity, revealing distinct scaling regimes ($\mathcal{O}\left(\sqrt{n}\right)$ and $\mathcal{O}\left(n\right)$) that emerge from correlation structures between vision representations independent of model size considerations. Second, the methodological approach differs substantially: Li et al. (2024c) conducts empirical experiments across various model configurations to derive scaling relationships through curve fitting, whereas this research develops a mathematical framework based on representational distance analysis that measures how models distinguish between vision-referencing sequences with shared prefix tokens. Third, the theoretical foundation diverges significantly: their work derives scaling relationships from observed experimental data patterns, while this research establishes mathematical descriptions based on the statistical properties of vision token representations and connects these properties to scaling behavior through divergence analysis of hidden state correlations.

While the existing literature demonstrates significant progress in vision token optimization and practical compression techniques, this work provides a systematic theoretical framework for understanding the fundamental relationship between the number of vision tokens and model performance. Unlike empirical approaches, the mathematical analysis is possible to establishes a relatively general principles.

3. Divergence Token Analysis

3.1 Motivation

In this section, the relationship between the number of newly introduced vision tokens and model performance on text generation given text prompts and vision inputs is investigated. However, directly measuring performance presents significant challenges, as it typically requires evaluation on specific datasets and may not generalize across different contexts. To address this limitation, this problem is reformulated using a more tractable approach.

Instead of directly measuring performance, the model's sensitivity to input variations is analyzed. This approach builds on a fundamental property of autoregressive models: under deterministic generation settings, for example random sampling is disabled or greedy decoding is used, identical inputs produce identical outputs. This deterministic behavior occurs because the model selects each token based on maximum likelihood estimation given the input prefix, eliminating stochastic variation.

Given this deterministic property, it is possible to examine how models respond to systematically varied inputs. Specifically, pairs of inputs that differ only in their suffix tokens are considered. The central question becomes: *how many suffix tokens are needed before the model produces noticeably different outputs?*

The intuition behind this approach is straightforward. When only one suffix token differs between two inputs, the differential information is limited, making it difficult for the model to distinguish between them. After more and more suffix tokens are introduced, the model gains additional information to differentiate the inputs, leading to increasingly divergent outputs. Eventually, with sufficient differential information, the model should easily distinguish between the inputs and produce substantially different responses.

This sensitivity analysis serves as a proxy for model capability. Models with better representational capacity should demonstrate appropriate sensitivity, detecting meaningful differences while maintaining stability for minor variations. The relationship between vision token quantity and this sensitivity profile provides insights into how additional vision tokens contribute to model performance.

This approach offers several advantages: it requires no dataset-specific evaluation, provides a more general framework applicable across different contexts, and directly measures the model's discriminative capacity. By analyzing how the introduction of vision tokens affects input sensitivity, it can help infer their impact on model performance without the complications of directly evaluation.

3.2 Input Pattern Formalization

To simplify the sensitivity analysis systematically, a uniform representation is required. Let $\langle|\text{txt}|\rangle$ denote text tokens and $\langle|\text{vis}|\rangle$ represent vision tokens. Input sequences in vision language model typically exhibit three fundamental patterns:

1. $\langle|\text{txt}|\rangle \ldots \langle|\text{txt}|\rangle \, \underline{\langle|\text{vis}|\rangle \ldots \langle|\text{vis}|\rangle}$: Vision tokens follow text tokens (e.g., "What is in the image? $\langle|\text{vis}|\rangle \ldots$")
2. $\langle|\text{txt}|\rangle \ldots \langle|\text{txt}|\rangle \, \underline{\langle|\text{vis}|\rangle \ldots \langle|\text{vis}|\rangle} \, \langle|\text{txt}|\rangle \ldots \langle|\text{txt}|\rangle$: Vision tokens are surrounded by text (e.g., "There is an image $\langle|\text{vis}|\rangle \ldots \langle|\text{vis}|\rangle$ Please describe it.")
3. $\underline{\langle|\text{vis}|\rangle \ldots \langle|\text{vis}|\rangle} \, \langle|\text{txt}|\rangle \ldots \langle|\text{txt}|\rangle$: Vision tokens precede text (e.g., "$\langle|\text{vis}|\rangle \ldots \langle|\text{vis}|\rangle$ What is in the image?")

These patterns can be unified through logical decomposition. Pattern 1 partitions into prefix text tokens and suffix vision tokens, pattern 2 decomposes into prefix text tokens and a compound component of vision tokens with trailing text, and pattern 3 constitutes a special case with zero-length prefix text. This yields a uniform representation:

$$[\langle|\text{txt}|\rangle \ldots \langle|\text{txt}|\rangle] \, [\langle|\text{vis}|\rangle \ldots] \,. \tag{3}$$

This representation can be further refined by systematically separate the prefix text tokens based on their relationship to visual content. The first category encompasses purely instructional content without visual grounding. Examples include phrases such as "Describe this image." or "What is this?" which serve as structural prompts that remain meaningful independent of visual content. The second category contains content that explicitly anchors linguistic elements to visual features. For instance, "What is in the box?", where "What is in the" does not refer to any specific visual element, while "box?" could be directly referenced to the visual content. Through this separation, the uniform representation can be further updated to:

$$\underbrace{[\langle|\text{txt}|\rangle \ldots \langle|\text{txt}|\rangle]}_{\text{non-referential}} \overbrace{[\langle|\text{txt}|\rangle \ldots \langle|\text{txt}|\rangle] \, [\langle|\text{vis}|\rangle \ldots]}^{\text{vision-referencing}} \,. \tag{4}$$

Building upon the uniform input pattern formalization, a formal mathematical framework is introduced to analyze token sensitivity. Without loss of generality, it can be assumed that non-referential prefix text tokens remain constant across input variations, while only the vision-referencing components differ between test cases. Let $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k\}$ denote the sequence of non-referential prefix text tokens that remain identical across input variations.

Suppose there are two distinct vision-referencing components, $\{\mathbf{a}_{k+1}, \mathbf{a}_{k+2}, \ldots, \mathbf{a}_{k+n}\}$ and $\{\mathbf{b}_{k+1}, \mathbf{b}_{k+2}, \ldots, \mathbf{b}_{k+n}\}$. The input sequences can then be expressed as:

$$\mathbf{x}_1 \quad \mathbf{x}_2 \quad \ldots \quad \mathbf{x}_k \quad \begin{cases} \mathbf{a}_{k+1} & \mathbf{a}_{k+2} & \ldots & \mathbf{a}_{k+n} \\ \mathbf{b}_{k+1} & \mathbf{b}_{k+2} & \ldots & \mathbf{b}_{k+n} \end{cases} \tag{5}$$

The divergence effect amplifies progressively as the number of differing tokens increases. When only one input token at position k differs ($\{\ldots, \mathbf{x}_k, \mathbf{a}_{k+1}\}$ and $\{\ldots, \mathbf{x}_k, \mathbf{b}_{k+1}\}$, $\mathbf{a}_{k+1} \neq \mathbf{b}_{k+1}$), the contextual discrepancy remains minimal. However, as more tokens differ ($\mathbf{a}_{k+1} \neq \mathbf{b}_{k+1}, \mathbf{a}_{k+2} \neq \mathbf{b}_{k+2}, \ldots$), the cumulative contextual differences compound at each generation step, leading to increasingly divergent output trajectories.

This phenomenon creates distinct *branches* in the output space. Each unique suffix sequence following the shared prefix $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k\}$ defines a separate trajectory in the model's generative process. The trajectory corresponding to suffix $\{\mathbf{a}_{k+1}, \mathbf{a}_{k+2}, \ldots, \mathbf{a}_{k+n}\}$ produces output branch $\{\mathbf{a}_{k+n+1}, \mathbf{a}_{k+n+2}, \ldots\}$, while suffix $\{\mathbf{b}_{k+1}, \mathbf{b}_{k+2}, \ldots, \mathbf{b}_{k+n}\}$ generates output branch $\{\mathbf{b}_{k+n+1}, \mathbf{b}_{k+n+2}, \ldots\}$. The autoregressive conditioning ensures that tokens generated at position $> k + n$ depend recursively on the diverging contexts, amplifying the initial differences.

Based on this framework, two sub research questions are formulated to guide the investigation under the theoretical perspective:

- **Metric definition:** How to quantify the distance between two input branches of $\{\ldots, \mathbf{x}_k, \mathbf{a}_{k+1}, \mathbf{a}_{k+2}, \ldots, \mathbf{a}_{k+n}\}$ and $\{\ldots, \mathbf{x}_k, \mathbf{b}_{k+1}, \mathbf{b}_{k+2}, \ldots, \mathbf{b}_{k+n}\}$? This question addresses the fundamental tool on which how to measure the divergence effect and assess the impact of input diversity.
- **Divergence scaling:** What is the functional relationship between the number of differing input tokens n and the distance between two input branches? Understanding this relationship provides insights into how the divergence effect scales with input diversity, which is crucial for forecasting the threshold value on significant divergence.

3.3 Metric Definition

In the following, the metric definition introduced in subsection 3.2 is further formalized to quantify the divergence between different computational branches in transformer-based language models.

Consider two input sequences with the pattern established in Equation (5). With the uniform input pattern formalization, it can be assumed that non-referential prefix text tokens remain constant across input variations, while only the vision-referencing components differ between test cases. Consider two input sequences with distinct vision-referencing components:

- Input sequence A: $\{\mathbf{x}_1, \ldots, \mathbf{x}_k, \mathbf{a}_{k+1}, \ldots, \mathbf{a}_{k+n}\}$,
- Input sequence B: $\{\mathbf{x}_1, \ldots, \mathbf{x}_k, \mathbf{b}_{k+1}, \ldots, \mathbf{b}_{k+n}\}$,

where, $\{\mathbf{x}_1, \ldots, \mathbf{x}_k\}$ constitutes the shared sequence of non-referential prefix text tokens that remain identical across input variations, while $\{\mathbf{a}_{k+1}, \ldots, \mathbf{a}_{k+n}\}$ and $\{\mathbf{b}_{k+1}, \ldots, \mathbf{b}_{k+n}\}$ represent the distinct vision-referencing components for sequences A and B, respectively.

In transformer-based large language models, the generation process is fundamentally linked to the hidden representations of the input sequence, specifically, the intermediate outputs of the model prior to the final linear projection layer. When analyzing divergent branches, these hidden representations can provide a more appropriate and informative metric than the raw input sequences themselves. Therefore, the hidden representations at each position in the sequence are considered as the primary object to be analyzed.

Since the prefix tokens $\{\mathbf{x}_1, \ldots, \mathbf{x}_k\}$ are identical in both sequences, their corresponding hidden representations are also identical. Consequently, the distance between these sequences should be zero up to position k, with divergence beginning at position $k+1$ where the suffixes start to differ. Denote $\mathbf{h}_i^{(A)}$ and $\mathbf{h}_i^{(B)}$ as the hidden representations at position $k+i$ for sequences A and B, respectively, and they are computed through the transformer architecture as:

$$\begin{aligned}\mathbf{h}_i^{(A)} &= \mathrm{LLM}(\mathbf{x}_1, \ldots, \mathbf{x}_k, \mathbf{a}_{k+1}, \ldots, \mathbf{a}_{k+i})[-1] \\ \mathbf{h}_i^{(B)} &= \mathrm{LLM}(\mathbf{x}_1, \ldots, \mathbf{x}_k, \mathbf{b}_{k+1}, \ldots, \mathbf{b}_{k+i})[-1]\end{aligned} \tag{6}$$

where LLM denotes the transformer model without the final linear projection layer, and $[-1]$ indicates the final hidden state. Each hidden representation $\mathbf{h}_i^{(A)}$ and $\mathbf{h}_i^{(B)}$ is computed by processing all preceding tokens from 0 to $k+i$.

The divergence metric between the two computational branches is formalized as a function of their respective hidden representation sequences:

$$\mathcal{D}(n) := \mathcal{D}\left\{\mathbf{h}_1^{(A)}, \ldots, \mathbf{h}_n^{(A)}; \mathbf{h}_1^{(B)}, \ldots, \mathbf{h}_n^{(B)}\right\} \tag{7}$$

where n denotes the number of differing tokens in the vision-referencing components, and $\mathcal{D}(n)$ quantifies the cumulative divergence between the two branches after processing n distinct tokens.

To develop an intuition of this distance metric, a geometric perspective is adopted, where hidden representations are considered as points in a high-dimensional space. The sequence progression can be conceptualized as a trajectory through this space, analogous to a reflectively random walk where each step represents the incorporation of a new token. Both branches begin at the same initial point (the shared prefix representation). As the sequences diverge through different vision-referencing components, each branch follows a distinct trajectory through the hidden space. The token at position $k+i$ moves the branch from its current position to a new location, creating a path that reflects the cumulative influence of all processed tokens.

The cumulative displacement between the two branches after n steps is captured by the sum of individual step differences:

$$\left(\sum_{i=1}^{n} \mathbf{h}_i^{(A)}\right) - \left(\sum_{i=1}^{n} \mathbf{h}_i^{(B)}\right) = \sum_{i=1}^{n} \left(\mathbf{h}_i^{(A)} - \mathbf{h}_i^{(B)}\right) \tag{8}$$

This motivates our computation of the distance between two branches, and the distance $\mathcal{D}(n)$ between the two branches after n steps can be expressed as:

$$\mathcal{D}(n) := \left\| \sum_{i=1}^{n} \left(\mathbf{h}_i^{(A)} - \mathbf{h}_i^{(B)}\right) \right\|_F , \tag{9}$$

where $\|\cdot\|_F$ represent the Frobenius norm. This cumulative distance metric captures the divergence between computational branches, providing a principled foundation for analyzing how different token sequences lead to distinct model behaviors.

3.4 Divergence Scaling

Subsection 3.3 established a distance metric. The subsequent investigation will explore how representational divergence scales with the number of differing vision-referencing tokens. This analysis is essential for two primary reasons: First, it reveals when statistically significant divergence occurs during trajectory separation. Second, it quantifies how input diversity propagates through the model's computational pathways, ultimately affecting output behaviors.

The foundation of our approach rests on the setting that distinct vision-referencing sequences generate unique computational trajectories in hidden space. Each suffix following the shared prefix creates a separate representational pathway. To derive generalizable insights beyond specific token sequences, an analysis over the expectation of various pairs of branches is considered to capture the model's general divergence behavior.

Let $\mathcal{V}$ denote the set of all vision-referencing sequence pairs that differ in at least one position, where input sequences are denoted $\{\ldots, \mathbf{a}_{k+1}, \ldots, \mathbf{a}_{k+n}\}$ and $\{\ldots, \mathbf{b}_{k+1}, \ldots, \mathbf{b}_{k+n}\}$. The expected divergence after n differing tokens is computed as:

$$\mathbb{E}_{\mathcal{V}}\left[\mathcal{D}(n)\right] = \mathbb{E}_{\mathcal{V}}\left[\left\|\sum_{i=1}^{n}\left(\mathbf{h}_i^{(A)} - \mathbf{h}_i^{(B)}\right)\right\|_F\right]. \tag{10}$$

Based on the settings, if $n \leq 0$, the sequence pairs are identical, and the divergence should be zero, $\mathbb{E}_{\mathcal{V}}\left[\mathcal{D}(n)\right] = 0$. Hence, the expectation is only meaningful for integer $n \geq 1$. The expectation in Equation (10) can be further bounded as:

$$\mathbb{E}_{\mathcal{V}}\left[\left\|\sum_{i=1}^{n}\left(\mathbf{h}_i^{(A)} - \mathbf{h}_i^{(B)}\right)\right\|_F\right] \leq \left(\mathbb{E}_{\mathcal{V}}\left[\left\|\sum_{i=1}^{n}\left(\mathbf{h}_i^{(A)} - \mathbf{h}_i^{(B)}\right)\right\|_F^2\right]\right)^{1/2}. \tag{11}$$

The expectation of the squared Frobenius norm can be decomposed into diagonal ($i = j$) and cross ($i \neq j$) terms, revealing the contribution of both individual variances and cross-correlations. The expansion of the expectation in Inequality (11) can be expressed as:

$$\begin{aligned}\mathbb{E}_{\mathcal{V}}\left[\left\|\sum_{i=1}^{n}\left(\mathbf{h}_i^{(A)} - \mathbf{h}_i^{(B)}\right)\right\|_F^2\right] &= \sum_{i=1}^{n}\mathbb{E}_{\mathcal{V}}\left[\left\|\mathbf{h}_i^{(A)} - \mathbf{h}_i^{(B)}\right\|_F^2\right] \\ &\quad + \sum_{i,j=1,i\neq j}^{n}\mathbb{E}_{\mathcal{V}}\left[\left(\mathbf{h}_i^{(A)} - \mathbf{h}_i^{(B)}\right)^\top\left(\mathbf{h}_j^{(A)} - \mathbf{h}_j^{(B)}\right)\right].\end{aligned} \tag{12}$$

The analysis of expected divergence should carefully consider the dependencies inherent in transformer-based sequence models. These dependencies manifest in two distinct but interconnected ways.

Intra-sequence dependencies: For a sequence model, the hidden representation at position $k+i$ is computed based on all preceding tokens from position 0 to i. This autoregressive nature creates strong dependencies within each sequence, where each hidden representation relates to all previous hidden representations. Consequently, $\mathbf{h}_i^{(A)}$ should have some dependency on $\mathbf{h}_j^{(A)}$ for $1 \leq j < i$. This temporal dependency structure means that the hidden representations in a sequence cannot be treated as independent samples, and their statistical relationships must be explicitly modeled in our analysis.

Inter-branch dependencies: When processing branching sequences that share the same initial sequence, the hidden representations in two branches are likely to exhibit a degree of commonality, especially when the length of the branching sequence is relatively short. This cross-branch dependency arises from three primary sources: First, both branches share the same initial context $\{\mathbf{x}_1, \dots, \mathbf{x}_k\}$, ensuring that the hidden representations generated in the branches are influenced by the same underlying information. Second, both branches are processed by the same sequence model, which applies its knowledge and understanding uniformly across both computational pathways. Third, when the branching sequences $\{\mathbf{a}_{k+1}, \dots, \mathbf{a}_{k+i}\}$ and $\{\mathbf{b}_{k+1}, \dots, \mathbf{b}_{k+i}\}$ are short, the new information introduced is relatively limited, which further reduces the divergence between the hidden representations. Consequently, $\mathbf{h}_i^{(A)}$ and $\mathbf{h}_i^{(B)}$ exhibit correlation that decreases as i increases, but remains significant for small values of i.

Considering these dependencies, the right-hand side (RHS) of Equation (12) becomes,

$$\begin{aligned}(12)\ \text{RHS} \leq\ & 2M^2 \sum_{i=1}^{n} \left(1 - \mathbb{E}_{\mathcal{V}}\left[\cos\left(\Theta_{ii}^{(AB)}\right)\right]\right) \\ & + M^2 \sum_{i,j=1, i\neq j}^{n} \left(\mathbb{E}_{\mathcal{V}}\left[\cos\left(\Theta_{ij}^{(AA)}\right)\right] + \mathbb{E}_{\mathcal{V}}\left[\cos\left(\Theta_{ij}^{(BB)}\right)\right] - 2\mathbb{E}_{\mathcal{V}}\left[\cos\left(\Theta_{ij}^{(AB)}\right)\right]\right),\end{aligned} \tag{13}$$

where the scalar M is defined as the largest value of the norm on arbitrary hidden states:

$$M = \max\left(\mathbb{E}_{\mathcal{V}}\left[\left\|\mathbf{h}_i^{(A)}\right\|_F\right], \mathbb{E}_{\mathcal{V}}\left[\left\|\mathbf{h}_i^{(B)}\right\|_F\right]\right), \quad \text{for all } i \in \{1, \dots, n\}, \tag{14}$$

and Θ denotes angles between hidden representations. Specifically, $\Theta_{ii}^{(AB)}$ is the angle between $\mathbf{h}_i^{(A)}$ and $\mathbf{h}_i^{(B)}$; while $\Theta_{ij}^{(AA)}$ and $\Theta_{ij}^{(BB)}$ correspond to the angles between $\mathbf{h}_i^{(A)}, \mathbf{h}_j^{(A)}$ and $\mathbf{h}_i^{(B)}, \mathbf{h}_j^{(B)}$ respectively; $\Theta_{ij}^{(AB)}$ reflects the angle between $\mathbf{h}_i^{(A)}$ and $\mathbf{h}_j^{(B)}$. For detailed derivative, please refer to Appendix A.1, and for the distribution of cosines, please refer to Appendix A.4.

To quantify the dependencies, four dependency measures that capture the essential correlation patterns in transformer architectures are defined. These measures capture the complex interactions between hidden representations into bounded parameters that facilitate theoretical analysis while preserving the fundamental characteristics of sequence processing. The dependency measures are constructed based on cosine similarities between hidden representations, which naturally capture the directional relationships between vectors in high-dimensional space.

Definition 1 (Dependency measures) *The dependency measures for sequence length n are defined as:*

$$
\begin{aligned}
\psi_{equal}^{(AB)}(n) &= \max\left\{0, \sup_{i=1,\ldots,n} \mathbb{E}_{\mathcal{V}}\left[\cos\left(\Theta_{ii}^{(AB)}\right)\right]\right\}, \\
\psi_{cross}^{(AB)}(n) &= \max\left\{0, \sup_{i,j=1,\ldots,n;i\neq j} \mathbb{E}_{\mathcal{V}}\left[\cos\left(\Theta_{ij}^{(AB)}\right)\right]\right\}, \\
\psi_{cross}^{(AA)}(n) &= \max\left\{0, \sup_{i,j=1,\ldots,n;i\neq j} \mathbb{E}_{\mathcal{V}}\left[\cos\left(\Theta_{ij}^{(AA)}\right)\right]\right\}, \\
\psi_{cross}^{(BB)}(n) &= \max\left\{0, \sup_{i,j=1,\ldots,n;i\neq j} \mathbb{E}_{\mathcal{V}}\left[\cos\left(\Theta_{ij}^{(BB)}\right)\right]\right\},
\end{aligned}
\tag{15}
$$

where:

- $\psi_{equal}^{(AB)}(n)$ *measures the same-position correlation between two branches, capturing how similar the representations remain at corresponding positions despite different branches (equal position, cross branches);*
- $\psi_{cross}^{(AB)}(n)$ *quantifies the cross-branch temporal correlation, measuring dependencies between representations at different positions across the two branches (cross positions, cross branches);*
- $\psi_{cross}^{(AA)}(n), \psi_{cross}^{(BB)}(n)$ *capture intra-sequence temporal dependencies within each branch, reflecting how similar the representations are at different positions within the same branch (cross positions, equal branch).*

The operator $\max\{0, \cdot\}$ takes only positive correlations to the dependency bounds, and the operator sup is used to capture the maximum correlation value over all possible hidden representations, capturing worst-case correlation scenarios for the theoretical bounds. Without loss of generality, it can be assumed that $\psi_{\text{cross}}^{(AA)}(n) = \psi_{\text{cross}}^{(BB)}(n)$ due to the symmetric nature of sequence branches, or can be considered as a replacement of parameters $\psi_{\text{cross}}^{(AA)}(n) \Leftarrow \frac{1}{2}\left(\psi_{\text{cross}}^{(AA)}(n) + \psi_{\text{cross}}^{(BB)}(n)\right)$.

With the dependency measures defined, and considering the symmetry of $\psi_{\text{cross}}^{(AA)}(n)$ and $\psi_{\text{cross}}^{(BB)}(n)$, right-hand side of Inequality (13) becomes,

$$
\begin{aligned}
(13)\text{ RHS} &\leq 2M^2 \sum_{i=1}^{n}\left(1 - \psi_{\text{equal}}^{(AB)}(n)\right) + M^2 \sum_{i,j=1,i\neq j}^{n}\left(\psi_{\text{cross}}^{(AA)}(n) + \psi_{\text{cross}}^{(BB)}(n) - 2\psi_{\text{cross}}^{(AB)}(n)\right) \\
&\leq 2M^2\left(n\left(1 - \psi_{\text{equal}}^{(AB)}(n)\right) + (n^2 - n)\left(\psi_{\text{cross}}^{(AA)}(n) - \psi_{\text{cross}}^{(AB)}(n)\right)\right).
\end{aligned}
\tag{16}
$$

For complete version of the derivative, please refer to Appendix A.1. To simply this notation, a shorthand function $\Upsilon(n)$ is introduced:

$$
\Upsilon(n) := n\left(1 - \psi_{\text{equal}}^{(AB)}(n)\right) + (n^2 - n)\left(\psi_{\text{cross}}^{(AA)}(n) - \psi_{\text{cross}}^{(AB)}(n)\right). \tag{17}
$$

Recalling Equation (10) and Inequality (11), the expected divergence after n differing tokens is bounded by:

$$\mathbb{E}_{\mathcal{V}}\left[\mathcal{D}(n)\right] = \mathcal{O}\left(\sqrt{n\left(1-\psi_{\text{equal}}^{(AB)}(n)\right)+(n^2-n)\left(\psi_{\text{cross}}^{(AA)}(n)-\psi_{\text{cross}}^{(AB)}(n)\right)}\right) := \mathcal{O}\left(\sqrt{\Upsilon(n)}\right), \tag{18}$$

where $\mathcal{O}\left(\cdot\right)$ represent the big $\mathcal{O}$ notation.

3.5 Divergence Scaling Properties

The asymptotic behavior of sequence divergence is critically determined by the relative magnitudes of the dependency measures $\psi_{\text{equal}}^{(AB)}(n)$, $\psi_{\text{cross}}^{(AA)}(n)$, and $\psi_{\text{cross}}^{(AB)}(n)$. To facilitate an understanding of these scaling properties, the analysis is first conducted under the simplified assumption that the dependency measures remain invariant with respect to sequence length. For analytical tractability, the discrete sequence length n is approximated as a continuous variable in the subsequent derivation.

Consider the simplified scenario where all dependency measures are constants independent of n: $\psi_{\text{equal}}^{(AB)}(n) \to \psi_{\text{equal}}^{(AB)}$, $\psi_{\text{cross}}^{(AA)}(n) \to \psi_{\text{cross}}^{(AA)}$, and $\psi_{\text{cross}}^{(AB)}(n) \to \psi_{\text{cross}}^{(AB)}$. In the following, such symbol with (n) stands for its function version, and that without (n) stands for its constant version. From Inequality (18), the argument of the square root can be decomposed into two components:

$$\begin{aligned} \text{Linear:} \quad & n\left(1-\psi_{\text{equal}}^{(AB)}-\psi_{\text{cross}}^{(AA)}+\psi_{\text{cross}}^{(AB)}\right), \\ \text{Quadratic:} \quad & n^2\left(\psi_{\text{cross}}^{(AA)}-\psi_{\text{cross}}^{(AB)}\right). \end{aligned} \tag{19}$$

Asymptotic behaviors shift as n change. The asymptotic scaling behavior is determined by which component dominates, leading to fundamentally different growth patterns for sequence divergence. If $\psi_{\text{cross}}^{(AA)} = \psi_{\text{cross}}^{(AB)}$, the quadratic component vanishes entirely, yielding only the linear component and leading to a sublinear scaling $\mathcal{O}\left(\sqrt{n}\right)$. Otherwise, if $\psi_{\text{cross}}^{(AA)} - \psi_{\text{cross}}^{(AB)} > 0$, a ratio $\rho(n)$ of the linear coefficient to the quadratic coefficient is defined to better characterize witch component dominates,

$$\rho(n) = \frac{1-\psi_{\text{equal}}^{(AB)}-\psi_{\text{cross}}^{(AA)}+\psi_{\text{cross}}^{(AB)}}{n\left(\psi_{\text{cross}}^{(AA)}-\psi_{\text{cross}}^{(AB)}\right)}. \tag{20}$$

The unusual case $\psi_{\text{cross}}^{(AA)} - \psi_{\text{cross}}^{(AB)} < 0$ is excluded here, since the quadratic component would result in a negative value that violates the positive constrains of the bound (please refer to Appendix A.2 for detailed justification). A critical balance point can be reached at $\rho(n) = 1$ when the linear and quadratic components contribute equally, yielding,

$$n|_{\rho(n)=1} = \frac{1-\psi_{\text{equal}}^{(AB)}}{\psi_{\text{cross}}^{(AA)}-\psi_{\text{cross}}^{(AB)}} - 1. \tag{21}$$

By setting $\rho(n) > 1$, and $\rho(n) < 1$, the divergence bound exhibits different scaling behaviors,

$$\mathcal{O}\left(\sqrt{n}\right) \Leftarrow n < n|_{\rho(n)=1}, \qquad n > n|_{\rho(n)=1} \Rightarrow \mathcal{O}\left(n\right). \tag{22}$$

When n is small, the linear component dominates and the divergence bound approaches to a sublinear scaling $\mathcal{O}\left(\sqrt{n}\right)$, indicating that sequence representations diverge slowly as the number of differing tokens increases. Conversely, when n is large, the quadratic component dominates and the bound degrades to a linear scaling $\mathcal{O}\left(n\right)$, suggesting rapid divergence growth. Under the linear scaling regime, a desire sequence divergence can be reached at a shorter sequence length compared to the quadratic scaling regime.

The transition between these scaling regimes can be understood through the lens of information availability and model confidence. When n is small, the distinct sequence branches are short, providing limited differential information between the sequences. Under these conditions, the model exhibits reduced confidence in distinguishing between the inputs, resulting in the sublinear scaling behavior $\mathcal{O}\left(\sqrt{n}\right)$ where divergence grows slowly. As n increases, the distinct sequence branches become longer, offering substantially more differential information. With access to this enriched information, the model gains increased capability to differentiate between the inputs, leading to the linear scaling regime $\mathcal{O}\left(n\right)$ where divergence exhibits rapid growth.

Influence of $\psi_{\textbf{cross}}^{(AA)} - \psi_{\textbf{cross}}^{(AB)}$ on divergence scaling. The quadratic component in Equation (19) is directly determined by the difference $\psi_{\text{cross}}^{(AA)} - \psi_{\text{cross}}^{(AB)}$. When this difference is large, the critical balance point $n|_{\rho(n)=1}$ from Equation (21) becomes smaller, resulting in a shorter sequence length required to achieve the same divergence threshold. Conversely, when this difference is small, $n|_{\rho(n)=1}$ increases, necessitating a longer sequence length to reach the same divergence threshold.

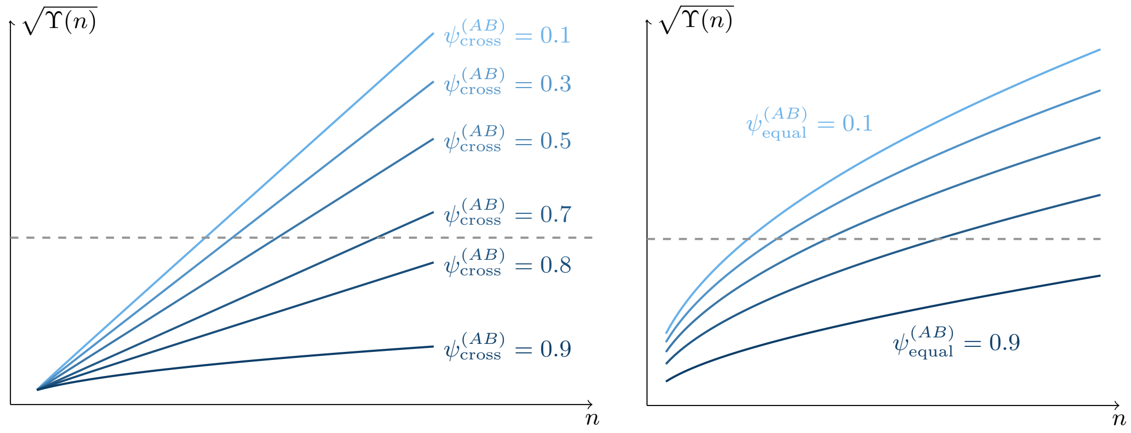
In practical terms, $\psi_{\text{cross}}^{(AA)}$ quantifies the temporal correlation within a single sequence branch, while $\psi_{\text{cross}}^{(AB)}$ measures the temporal correlation between two distinct sequence branches. The difference $\psi_{\text{cross}}^{(AA)} - \psi_{\text{cross}}^{(AB)}$ can be interpreted as the relative strength of intra-branch dependencies compared to cross-branch dependencies. When this difference is large, the hidden representations of the two sequence branches exhibit substantial dissimilarity, enabling the model to distinguish between them with greater ease. Consequently, fewer tokens are required to reach the desired divergence threshold.

Conversely, when this difference is small, the hidden representations of the two sequence branches remain highly similar, making it more challenging for the model to differentiate between them. Under such conditions, a longer sequence length is necessary to achieve the desired divergence threshold.

To provide an illustration of this relationship, $\psi_{\text{cross}}^{(AA)}$ and $\psi_{\text{equal}}^{(AB)}$ are held constant while $\psi_{\text{cross}}^{(AB)}$ is varied systematically, as demonstrated in Figure 1a. The results align with the theoretical analysis: when $\psi_{\text{cross}}^{(AB)}$ is small, the difference $\psi_{\text{cross}}^{(AA)} - \psi_{\text{cross}}^{(AB)}$ becomes large, and the figure confirms that a smaller value of n is sufficient to reach the desired threshold.

Influence of $\psi_{\textbf{equal}}^{(AB)}$ on divergence scaling. From Equation (19), the linear component is governed by the term $1 - \psi_{\text{equal}}^{(AB)}$. When this difference is large, the critical balance point $n|_{\rho(n)=1}$ from Equation (21) increases accordingly.

The influence of $\psi_{\text{equal}}^{(AB)}$ on the scaling behavior exhibits two distinct characteristics. First, the parameter has limited impact on the balance point since $1 - \psi_{\text{equal}}^{(AB)}$ appears in the numerator of $n|_{\rho(n)=1}$, making its influence weaker than that of the quadratic coefficient $\psi_{\text{cross}}^{(AA)} - \psi_{\text{cross}}^{(AB)}$ in the denominator. The balance point varies linearly with $\psi_{\text{equal}}^{(AB)}$, and since



(a) (b)

Figure 1: (a) $\Upsilon(n)$ with respect to $\psi^{(AB)}_{\text{cross}}$ with fixed $\psi^{(AB)}_{\text{equal}} = 0.7$ and $\psi^{(AA)}_{\text{cross}} = 0.9$. Lower values of $\psi^{(AB)}_{\text{cross}}$ lead to faster divergence growth, enabling threshold crossing at shorter sequence lengths. (b) $\Upsilon(n)$ with respect to $\psi^{(AB)}_{\text{equal}}$ with fixed $\psi^{(AB)}_{\text{cross}} = 0.899$ and $\psi^{(AA)}_{\text{cross}} = 0.900$. Higher values of $\psi^{(AB)}_{\text{equal}}$ lead to smaller values of $\Upsilon(n)$.

$\psi^{(AB)}_{\text{equal}}$ is constrained to the interval $[0, 1]$, its impact on $n|_{\rho(n)=1}$ remains bounded. Second, the parameter exerts direct influence on the linear component by primarily affecting the linear term in $\Upsilon(n)$. When $\psi^{(AB)}_{\text{equal}}$ is smaller, the coefficient $1 - \psi^{(AB)}_{\text{equal}}$ becomes larger, resulting in a larger linear component and consequently faster approach to the desired divergence threshold.

The practical impact of $\psi^{(AB)}_{\text{equal}}$ depends critically on which scaling regime dominates. When the quadratic term is dominant (large n), the influence of $\psi^{(AB)}_{\text{equal}}$ becomes negligible. However, when the linear term dominates (small n), $\psi^{(AB)}_{\text{equal}}$ exerts substantial influence on the divergence scaling behavior.

In practical applications, $\psi^{(AB)}_{\text{equal}}$ quantifies the similarity between hidden representations of two distinct sequence branches at corresponding positions in the input sequences. When this value is large, the representations at corresponding positions across branches remain highly similar, making it more challenging for the model to differentiate between the sequences. Consequently, during the initial processing phase (small n), when $\psi^{(AB)}_{\text{equal}}$ is small, the model can readily distinguish between the two branches from the outset. This early discrimination capability enables the model to reach the desired divergence threshold more rapidly.

Figure 1b illustrates this relationship. As $\psi^{(AB)}_{\text{equal}}$ decreases, the divergence threshold is reached more quickly for a given sequence length.

Connection to scaling behavior. Let $S(n)$ denote the model's performance on a given task after processing n differing tokens. The performance is hypothesized to be proportional to the expected divergence:

$$S(n) \propto \left(\mathbb{E}_{\mathcal{V}}\left[\mathcal{D}(n)\right]\right)^{\beta} \propto \left(\sqrt{\Upsilon(n)}\right)^{\beta}, \tag{23}$$

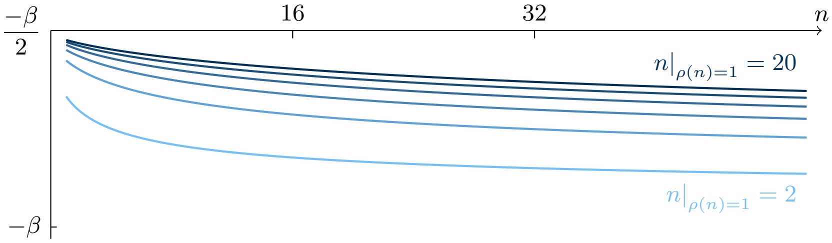


Figure 2: $\alpha(n)$ with $n > 1$ and $\psi_{\text{cross}}^{(AA)} - \psi_{\text{cross}}^{(AB)} > 0$.

where $\beta > 0$ is a constant that characterizes the relationship between performance and divergence. For instance, when $\beta = 1$, the performance $S(n)$ scales linearly with $\mathbb{E}_{\mathcal{V}}\left[\mathcal{D}(n)\right]$.

Following the scaling formulation introduced in Kaplan et al. (2020); Hoffmann et al. (2022), the relationship between performance and sequence length is considered to be

$$S(n) \approx \frac{c}{n^{\alpha(n)}}, \tag{24}$$

where c is the scaling constant, and $\alpha(n)$ is the scaling exponent. Consequently, it is natural to establish the scaling relationship

$$\gamma \Upsilon(n)^{\beta/2} = \frac{c}{n^{\alpha(n)}}, \tag{25}$$

which connects the theoretical divergence bound to empirical scaling behavior. The value γ is the scaling factor for matching $S(n)$ and $\Upsilon(n)^{\beta/2}$. With the constant setting, three main results are summarized as below (Please refer to Appendix A.3 for the detailed discussion). The normalization constant can be set as

$$c = \gamma \left(1 - \psi_{\text{equal}}^{(AB)}\right)^{\frac{\beta}{2}}. \tag{26}$$

When $n > 1$ and $\psi_{\text{cross}}^{(AA)} - \psi_{\text{cross}}^{(AB)} = 0$, the scaling exponent is given by,

$$\alpha(n) = -\frac{\beta}{2}. \tag{27}$$

When $n > 1$ and $\psi_{\text{cross}}^{(AA)} - \psi_{\text{cross}}^{(AB)} > 0$, the scaling exponent becomes,

$$\alpha(n) = \frac{\beta}{2} \log_n \left(\frac{1 + n|_{\rho(n)=1}}{n + n|_{\rho(n)=1}} \right) - \frac{\beta}{2}. \tag{28}$$

This $\alpha(n)$ is a monotonically decreasing function with respect to n and will approach to $-\beta$ as n increases. In order to provide a more intuitive understanding, the curves of $\alpha(n)$ under different values of $n|_{\rho(n)=1}$ are shown in Figure 2, which shows the slope of the curves becomes stable as n increases. Therefore, it is reasonable to simplify the expressions as a constant value $\alpha(n) \to \alpha$ in the following estimations and evaluations.

Smaller values of $\alpha(n)$ (corresponding to larger values of $|\alpha(n)|$) lead to faster growth in performance $S(n)$. Under the condition $n > 1$ and $\psi_{\text{cross}}^{(AA)} - \psi_{\text{cross}}^{(AB)} > 0$, for a fixed sequence length n, a smaller balance point $n|_{\rho(n)=1}$ results in a correspondingly smaller value of $\alpha(n)$. This relationship also aligns with the observations from the upper bound $\mathcal{O}\left(\sqrt{\Upsilon(n)}\right)$. A smaller balance point indicates that fewer tokens n are required to transition into the linear scaling regime $\mathcal{O}(n)$, such that fewer tokens n can lead to a larger bound.

These expressions reveal how the scaling exponent $\alpha(n)$ depends on the dependency measures, providing a direct link between the theoretical analysis of sequence divergence and the empirical observations of model performance scaling.

3.6 Empirical Estimation

The theoretical analysis of divergence between branching sequences has been established in the preceding subsections. This subsection provides empirical validation of the theoretical framework through the estimation of the expected divergence $\mathbb{E}_{\mathcal{V}}[\mathcal{D}(n)]$ and the dependency measures $\psi^{(AB)}_{\text{equal}}(n)$, $\psi^{(AA)}_{\text{cross}}(n)$, and $\psi^{(AB)}_{\text{cross}}(n)$ on pretrained large language models using a dataset of paired sequences. Through this empirical analysis, the validity of the theoretical predictions is demonstrated.

Dataset construction. To facilitate empirical estimation, a dataset of paired input sequences that align with our theoretical setting is required. Each pair consists of sequences that share an identical prefix but diverge in their vision-referencing components. Specifically, each sequence comprises three components: a system prompt, a query, and an answer. Within each pair, the system prompt and query remain identical (forming the shared prefix), while the answers differ, creating the branching sequences that correspond to branches A and B in the theoretical analysis.

The hidden representations are computed specifically based on the 'answer' component of the paired sequences. This approach enables focused analysis of how representational divergence propagates through the model's computational pathways as the branching sequences evolve. The dataset for Direct Preference Optimization (DPO) (Rafailov et al., 2023) matches the requirements perfectly. For this empirical analysis, dataset 'ocra-dpo-pairs' [1] is utilized. This dataset contains the necessary paired sequences with the required structure for our estimation procedure. The 'chosen' answer is the computed as branching sequence A, and the 'rejected' answer is the computed as branching sequence B.

Computational procedure. The empirical estimation follows a systematic four-step process:

1. *Tokenization.* The positive and negative sequences are converted to input token sequences $\{\mathbf{x}_1, \ldots, \mathbf{x}_k, \mathbf{a}_{k+1}, \ldots, \mathbf{a}_{k+n}\}$ and $\{\mathbf{x}_1, \ldots, \mathbf{x}_k, \mathbf{b}_{k+1}, \ldots, \mathbf{b}_{k+n}\}$, where k represents the length of the shared prefix.

2. *Hidden representation extraction.* The input tokens are processed through the LLM (excluding the final output layer) to obtain the hidden representations:

$$\mathbf{h}_i^{(A)} = \text{LLM}(\mathbf{x}_1, \ldots, \mathbf{x}_k, \mathbf{a}_{k+1}, \ldots, \mathbf{a}_{k+i})[-1],$$
$$\mathbf{h}_i^{(B)} = \text{LLM}(\mathbf{x}_1, \ldots, \mathbf{x}_k, \mathbf{b}_{k+1}, \ldots, \mathbf{b}_{k+i})[-1].$$

---

[1] `https://huggingface.co/datasets/Intel/orca_dpo_pairs`

3. *Statistical estimation.* The empirical estimates are computed across multiple data samples to ensure statistical reliability and robustness of the results.

Empirical estimation of $\mathbb{E}_{\mathcal{V}}\left[\mathcal{D}(n)\right]$. Following the definition in Equation (10), the expected divergence is computed as the expected Frobenius norm of the cumulative hidden representation differences. The estimation is performed by systematically enumerating each sequence length n and computing the corresponding average Frobenius norm across the vocabulary distribution. For instance, the expected divergences for $n = 1$ and $n = 2$ are estimated as:

$$\mathbb{E}_{\mathcal{V}}\left[\mathcal{D}(1)\right] \Leftarrow \mathbb{E}_{\mathcal{V}}\left[\left\|\mathbf{h}_1^{(A)} - \mathbf{h}_1^{(B)}\right\|_F\right], \mathbb{E}_{\mathcal{V}}\left[\mathcal{D}(2)\right] \Leftarrow \mathbb{E}_{\mathcal{V}}\left[\left\|\mathbf{h}_1^{(A)} - \mathbf{h}_1^{(B)}\right\|_F + \left\|\mathbf{h}_2^{(A)} - \mathbf{h}_2^{(B)}\right\|_F\right].$$

For the expectation $\mathbb{E}_{\mathcal{V}}$, it is considered as an empirical average,

$$\mathbb{E}_{\mathcal{V}}\left[\mathcal{D}(n)\right] \Leftarrow \frac{1}{N}\sum_{i=1}^{N}\mathcal{D}_i(n),$$

where N is the number of samples and $\mathcal{D}_i(n)$ is the i-th sample of the expected divergence for sequence length n.

Empirical estimation of dependency measures. The four dependency measures introduced in Definition 1 require empirical estimation from the data. The same-position and cross-branch temporal correlations are computed as:

$$\psi_{\text{equal}}^{(AB)}(n) \Leftarrow \mathbb{E}_{\mathcal{V};i=1,\ldots,n}\cos\left(\Theta_{ii}^{(AB)}\right), \quad \psi_{\text{cross}}^{(AB)}(n) \Leftarrow \mathbb{E}_{\mathcal{V};i,j=1,\ldots,n;i\neq j}\cos\left(\Theta_{ij}^{(AB)}\right). \tag{29}$$

Due to the inherent structure of the DPO dataset, where sequences are organized into distinct preference pairs, all 'chosen' responses are assigned to branch A, and all 'rejected' responses are assigned to branch B. This assignment introduces asymmetry between the sequence branches, violating the symmetry assumption $\psi_{\text{cross}}^{(AA)}(n) = \psi_{\text{cross}}^{(BB)}(n)$ from the theoretical framework.

To address this asymmetry and obtain unbiased estimates, the intra-branch temporal correlations are computed separately for each branch:

$$\psi_{\text{cross}}^{(++)}(n) \Leftarrow \mathbb{E}_{\mathcal{V};i,j=1,\ldots,n;i\neq j}\cos\left(\Theta_{ij}^{(AA)}\right), \quad \psi_{\text{cross}}^{(--)}(n) \Leftarrow \mathbb{E}_{\mathcal{V};i,j=1,\ldots,n;i\neq j}\cos\left(\Theta_{ij}^{(BB)}\right). \tag{30}$$

The final symmetric estimate is obtained by averaging the two branch-specific correlations:

$$\psi_{\text{cross}}^{(AA)}(n) = \psi_{\text{cross}}^{(BB)}(n) \Leftarrow \frac{1}{2}\left(\psi_{\text{cross}}^{(++)}(n) + \psi_{\text{cross}}^{(--)}(n)\right). \tag{31}$$

This averaging procedure ensures that the theoretical symmetry assumption is preserved in the empirical analysis while accounting for the structural characteristics of the preference-based dataset.

Validation of the theoretical bound. Equation (18) establishes an upper bound for the expected divergence in terms of the dependency measures. To validate the theoretical bound against empirical observations, a scaling factor λ is introduced such that the expected divergence can be approximated as:

$$\mathbb{E}_{\mathcal{V}}\left[\mathcal{D}(n)\right] \approx \lambda\sqrt{\Upsilon(n)}. \tag{32}$$

Given that all necessary dependency measures have been empirically estimated, the function $\Upsilon(n)$ can be computed directly using the estimated parameters:

$$\Upsilon(n) \Leftarrow n\left(1-\psi_{\text{equal}}^{(AB)}(n)\right)+\left(n^2-n\right)\left(\psi_{\text{cross}}^{(AA)}(n)-\psi_{\text{cross}}^{(AB)}(n)\right). \tag{33}$$

The optimal scaling factor λ is determined by minimizing the squared error between the empirical divergence and the theoretical prediction:

$$\lambda = \underset{\lambda}{\operatorname{argmin}} \sum_n \left(\mathbb{E}_{\mathcal{V}}\left[\mathcal{D}(n)\right]-\lambda\sqrt{\Upsilon(n)}\right)^2. \tag{34}$$

Empirical results. The empirical estimations are taken under various of LLMs, including Vicuna-v1.5-7b (Chiang et al., 2023), Phi-3-mini (Abdin et al., 2024), Llama2-7b (Touvron et al., 2023), Llama-3-8B (Dubey et al., 2024), and Qwen series (Bai et al., 2023a).

Figure 3 presents the estimated dependency measures decomposed by response type. The measures $\psi_{\text{cross}}^{(++)}(n)$ and $\psi_{\text{cross}}^{(--)}(n)$ exhibit similar trends with respect to sequence length n, while their specific values differ across response categories. Different base models demonstrate distinct patterns for $\psi_{\text{cross}}^{(++)}(n)$ and $\psi_{\text{cross}}^{(--)}(n)$, indicating model-specific dependency characteristics.

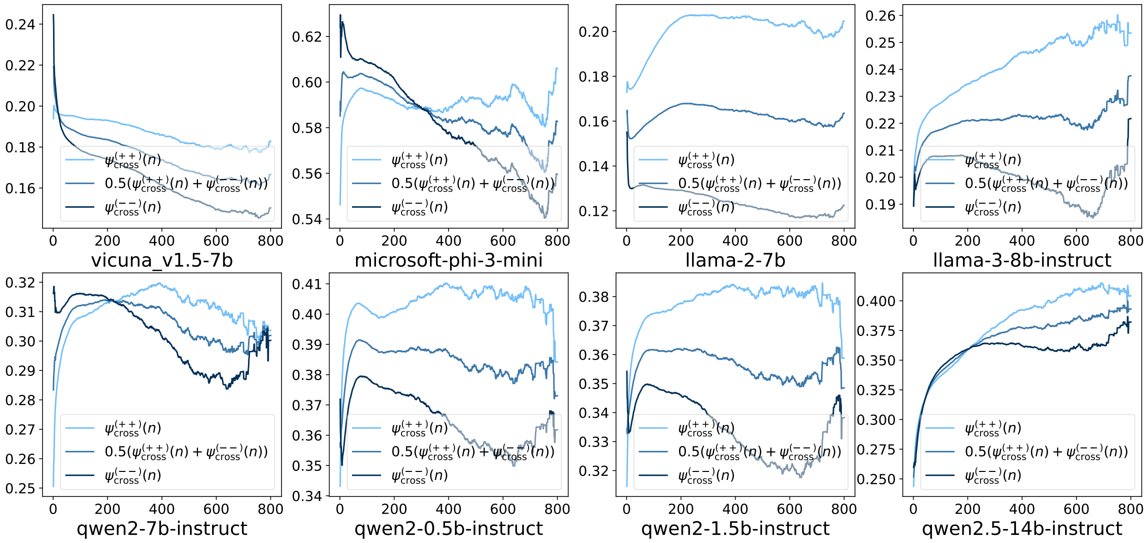


Figure 3: Estimated dependency measures: $\psi_{\text{cross}}^{(AA)}(n) = \frac{1}{2}\left(\psi_{\text{cross}}^{(++)}(n)+\psi_{\text{cross}}^{(--)}(n)\right)$, where $\psi_{\text{cross}}^{(++)}(n)$ corresponds to branches with 'chosen' responses and $\psi_{\text{cross}}^{(--)}(n)$ corresponds to branches with 'rejected' responses.

Figure 4 illustrates the estimated values of the three fundamental dependency measures: $\psi_{\text{cross}}^{(AA)}(n)$, $\psi_{\text{equal}}^{(AB)}(n)$, and $\psi_{\text{cross}}^{(AB)}(n)$. The evolution patterns observed across different base models exhibit similar characteristics, beginning with an initial transient phase and rapidly converging towards a stable regime. The variation range from minimal to maximal values remains constrained within approximately 0.2 across all base models. Consequently, these dependency measures can be reasonably approximated as constants for analytical purposes, provided that the approximation error introduced by this simplification is deemed acceptable for the intended analysis.

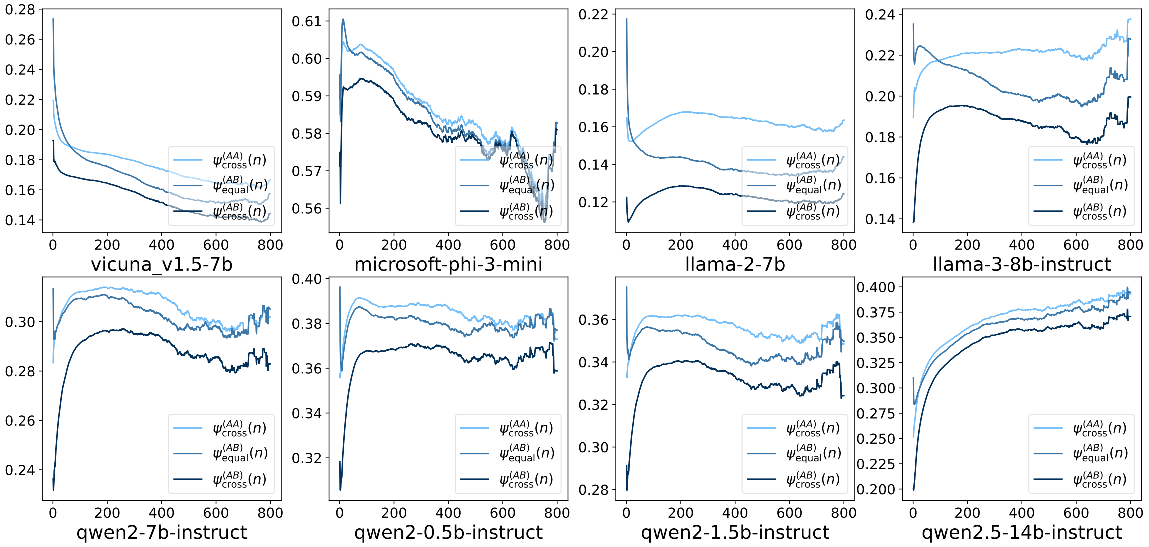


Figure 4: Estimated three dependency measures: $\psi^{(AA)}_{\text{cross}}(n)$ (intra-sequence temporal correlation), $\psi^{(AB)}_{\text{equal}}(n)$ (same-position cross-branch correlation), and $\psi^{(AB)}_{\text{cross}}(n)$ (cross-branch temporal correlation).

With all necessary dependency measures estimated, the function $\Upsilon(n)$ can be computed directly using Equation (17). Combined with the estimation of $\mathbb{E}_{\mathcal{V}}[\mathcal{D}(n)]$ and the scaling parameter λ from Equation (34), the theoretical predictions can be validated against empirical observations, as shown in Figure 5. The results demonstrate that the computed curves of $\lambda\sqrt{\Upsilon(n)}$ closely align with the empirical observations of $\mathbb{E}_{\mathcal{V}}[\mathcal{D}(n)]$, providing empirical support for the theoretical framework.

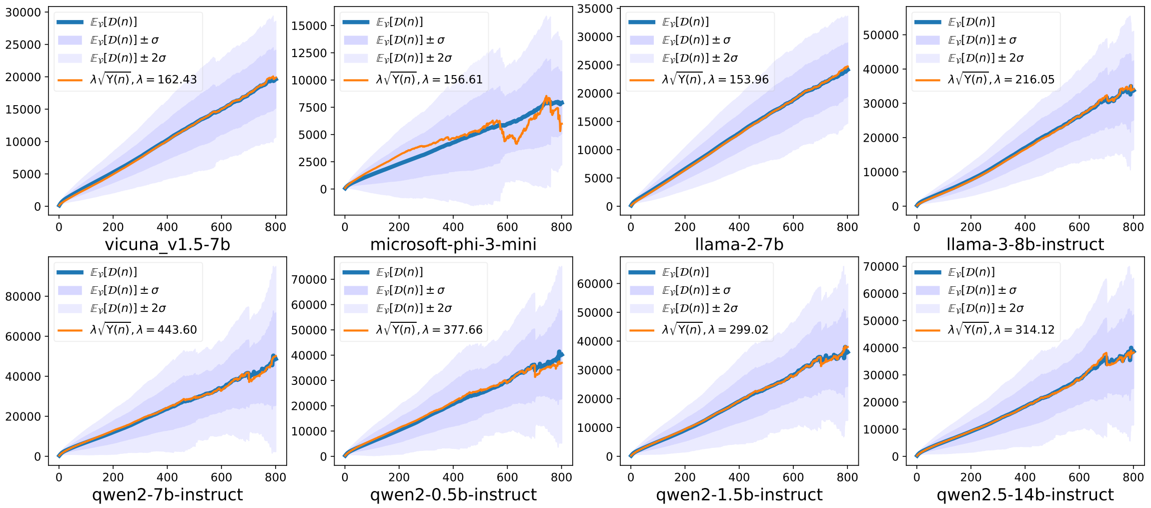
4. Model Design

4.1 Motivation

Section 3 provides a theoretical insight into vision token scaling behavior. This section presents a vision language model architecture designed to systematically investigate the relationship between number of vision tokens and model performance, providing a controlled experimental setting for empirical validation.

The theoretical analysis establishes that performance scales as $S(n) \approx c/n^{\alpha(n)}$, where n represents the number of differing vision-referencing tokens. To empirically validate this scaling relationship, three architectural requirements emerge:

High-resolution image support. To provide sufficient visual information for discriminative analysis and enable systematic variation of the number of vision tokens, the model should be able to process high-resolution images that generate rich initial vision sequences. This ensures adequate visual detail while allowing flexible token count selection for scaling experiments.



vicuna_v1.5-7b
microsoft-phi-3-mini
llama-2-7b
llama-3-8b-instruct
qwen2-7b-instruct
qwen2-0.5b-instruct
qwen2-1.5b-instruct
qwen2.5-14b-instruct

Figure 5: Comparison between empirical observations and theoretical predictions: estimated mean and standard deviation of $\mathbb{E}_{\mathcal{V}}\left[\mathcal{D}(n)\right]$ alongside computed curves of $\lambda\sqrt{\Upsilon(n)}$. The close alignment validates the theoretical framework.

Controllable number of vision tokens. Systematic investigation of the scaling relationship requires precise control over the number of vision tokens n fed into the language model. The model should produce a fixed number of fused vision tokens, enabling controlled experimentation across different token counts.

Question-conditioned vision processing. The uniform input pattern formalization in Equation (4) distinguishes between non-referential text tokens and vision-referencing components. This separation suggests that text tokens can guide vision token selection and processing. By conditioning vision processing on question semantics, it is possible that the model can optimize the vision-referencing component, potentially improving the scaling efficiency.

The resulting architecture implements these principles through a fusion module that processes vision tokens conditioned on text questions, producing controllable numbers of fused tokens for systematic scaling analysis.

4.2 Model Architecture

The proposed model architecture comprises three primary components: a vision encoder, a fusion module, and a large language model backbone, as illustrated in Figure 6. Each component addresses specific requirements derived from the theoretical analysis while enabling systematic investigation of vision token scaling behavior.

Vision encoder. The vision encoder processes input images to generate vision tokens that encapsulate visual information. To generate sufficient vision tokens for scaling experiments, the high resolution image support requirement is addressed by employing a dual-view preprocessing strategy, inspired by Dong et al. (2024). Input images are transformed into one global and several local views: the global view provides holistic context through padding and zooming to match the target resolution, while local views capture fine-grained details by

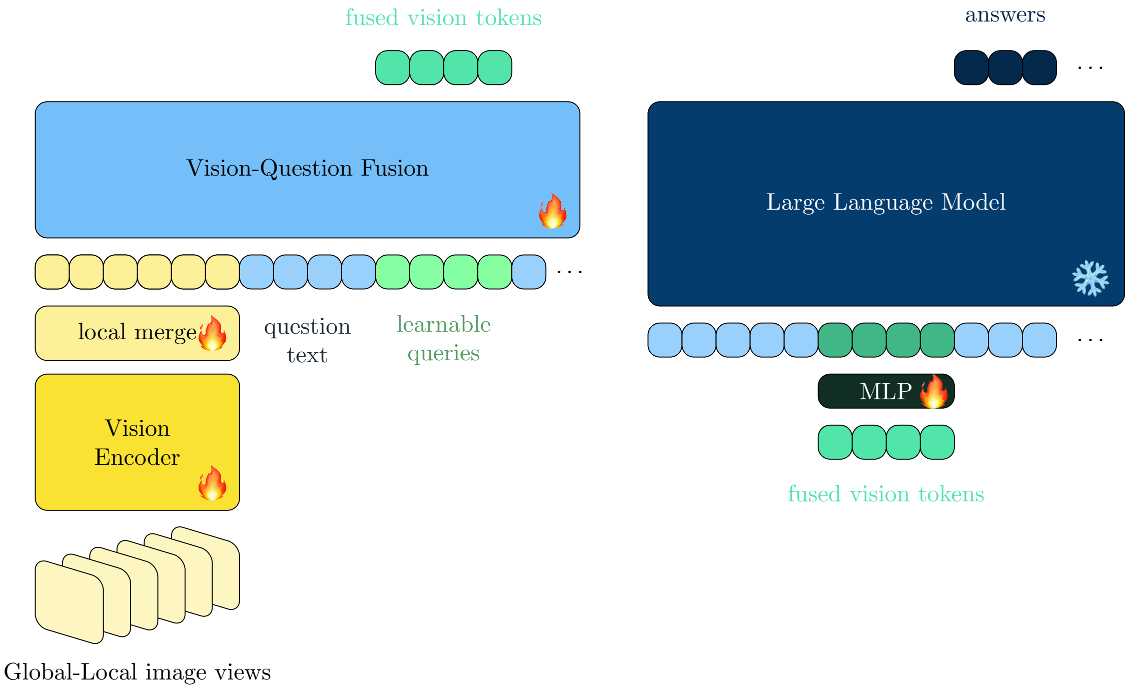


Figure 6: The architecture of the proposed model. Vision tokens are fused with question text and learnable queries before being processed by the LLM. Modules marked with 'fire' are updated during fine-tuning, whereas modules marked with 'snowflake' remain frozen.

dividing the processed image into smaller patches. This approach enables the vision encoder pretrained on smaller resolutions to process high resolution images effectively. For example, the pretrained CLIP ViT-H/14 encoder (Radford et al., 2021) can be used to process high resolution content beyond its native $(336, 336)$ resolution limit.

To efficiently manage the large number of vision tokens generated during the global-local view preprocessing, a local merges strategy is employed, wherein multiple vision tokens, such as four neighboring tokens, are aggregated into a single token, thereby reducing the token count by a factor of four and streamlining the subsequent processing stages. This preprocessing methodology enables the vision encoder to process high resolution images without any adaptations required to its underlying architecture.

Fusion module. The fusion module serves two distinct requirements: controlling the number of vision tokens and integrating vision tokens with text tokens from questions. For token count control, the module incorporates learnable queries inspired by Q-Former (Li et al., 2023b) that interact with vision tokens to selectively attend to relevant visual information, distilling a fixed number of output tokens for systematic scaling analysis. For integration with text questions, the module employs a lightweight decoder-only self-attention transformer architecture following the structure of Llama2 (Touvron et al., 2023), providing a full interaction of vision tokens, question text tokens and the learnable queries.

Language model backbone. The large language model backbone pretrained exclusively on text data is extended to process multimodal input by concatenating the text tokens with the fused vision tokens from the fusion module. To ensure proper alignment between the fused vision tokens and the text tokens, a projection layer is applied to match their dimensional compatibility and aligned two modalities.

Fine-tuning. The fusion module, vision encoder, projection layers, and the large language model backbone are joined to build the main architecture. During fine-tuning, the fusion module, vision encoder, and some other projection layers are updated to adapt to the integrated multimodal input, enabling the model to generate coherent, context-aware responses. The large language model backbone remains frozen. This approach allows for isolating the effects of vision token scaling from changes in language modeling capacity, enabling precise analysis of the scaling relationship established in our theoretical framework.

4.3 Formats of Input Sequences

Two distinct token sequence formats are introduced in this subsection. The first format represents the input to the fusion module, while the second format pertains to the input to the large language model.

Input format of fusion module. Three distinct tokens are concatenated to form the input to the fusion module. Vision tokens, generated by the vision encoder, encapsulate visual features extracted from the input image. Text tokens are derived from the question text and mapped through the embedding layer. Learnable queries are learnable codebook designed to facilitate selective attention within the fusion module. Figure 7 illustrates an example of this format. This format enables efficient processing of multiple question-answer pairs within a single forward pass.

To optimize computational efficiency, the fusion module employs an asymmetric allocation strategy. The first question is paired with n_l learnable queries, while subsequent questions are each paired with n_s learnable queries, where $n_s < n_l$. This design addresses two fundamental considerations. First, questions within a multi-question context often share common visual information that can be encoded once and reused. The initial n_l learnable queries capture this shared contextual information and general visual features that benefit all subsequent questions. The remaining questions, a shorter n_s query length, only encoding question-specific information. This design efficiently balances the inclusion of shared and question-specific information while reducing redundancy. Secondly, this approach significantly reduces the total number of learnable queries, as well as the corresponding fused vision tokens. For K questions, using n_l learnable queries for all questions would result in $K \times n_l$ learnable queries. In contrast, by employing n_l learnable queries after the first question and n_s learnable queries after the remaining questions, the total number of tokens is reduced to $n_l + K \times n_s$, where $n_s < n_l$. This reduction can enhance computational efficiency and scalability.

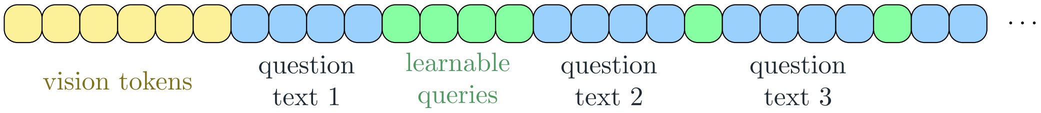


Figure 7: Example of the input token sequence to the fusion module.

Input format of language model. Each large language model requires its specific input format to ensure effective processing. To incorporate the fused vision tokens into the LLM input, these tokens are positioned at the beginning of the input sequence for each question, while all other text structures remain unchanged. This arrangement ensures that the visual information is readily accessible to the model, providing a consistent context for interpreting the question-specific tokens that follow. An example of this input format is shown in Figure 7. The lengths of the fused vision tokens follow the same pattern as the learnable queries, with the first question using n_l tokens and the remaining questions using n_s tokens.

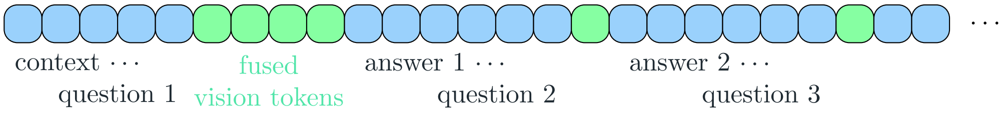


Figure 8: Example of the input token sequence to the fusion module. If needed, the fused vision token would be projected to match the shape of the LLM.

4.4 Loss Function

The fusion module, being randomly initialized, requires carefully designed training objectives to effectively integrate visual and textual information. A two-stage training strategy is employed, with contrastive loss for initial alignment and generation loss for answer production. The training pipeline involves two principal stages. Initially, preliminary training is conducted on the vision encoder, the fused model with the contrastive loss, and subsequently, fine-tuning is performed with the large language model backbone on generation loss.

Contrastive loss. The contrastive loss ensures semantic alignment between fused vision representations and text representations, following the CLIP framework. This objective enables the fusion module to produce vision representations that are semantically compatible with text representations from the CLIP text encoder.

The loss operates on ⟨|eos|⟩ tokens, which serve as boundary markers for question-answer groups. In multi-question scenarios, each question group (comprising questions and learnable queries) terminates with an ⟨|eos|⟩ token, as does each independently encoded answer. The contrastive loss specifically targets these ⟨|eos|⟩ tokens to enforce alignment between question group representations and their corresponding answers. For non-⟨|eos|⟩ tokens within the learnable queries, we apply an orthogonality constraint through identity loss. This encourages these tokens to encode distinct, complementary information, maximizing their representational diversity and capacity.

Let $\Omega_{\langle|\text{eos}|\rangle}$ denote the index set of ⟨|eos|⟩ tokens. For $i, j \in \Omega_{\langle|\text{eos}|\rangle}$, let $\mathbf{v}_i$ represent the fused vision token corresponding to the ⟨|eos|⟩ token for the i-th question-answer pair, and let $\mathbf{s}_j$ denote the answer text token associated with the ⟨|eos|⟩ token for the j-th question-answer pair. In addition, let $\overline{\Omega}_{\langle|\text{eos}|\rangle}$ represent the index set of non-⟨|eos|⟩ tokens within the fused vision tokens. For these tokens, define $\mathbf{Y}$ as a matrix where each row corresponds to a non-⟨|eos|⟩ token, normalized to have Frobenius norm of one. Formally, each row is defined as $\mathbf{Y}_i = \mathbf{y}_i / \|\mathbf{y}_i\|_F$, where $i \in \overline{\Omega}_{\langle|\text{eos}|\rangle}$.

The contrastive loss, $\mathcal{L}_{\text{cont}}$, is defined as:

$$\mathcal{L}_{\text{cont}} = \frac{-1}{|\Omega_{\langle|\text{eos}|\rangle}|} \sum_{k \in \Omega_{\langle|\text{eos}|\rangle}} \log \left(\frac{\exp\left(\mathbf{v}_k^\top \mathbf{s}_k\right)}{\sum_{i,j \in \Omega_{\langle|\text{eos}|\rangle}} \exp\left(\mathbf{v}_i^\top \mathbf{s}_j\right)} \right) + \frac{1}{|\overline{\Omega}_{\langle|\text{eos}|\rangle}|} \left\| \mathbf{Y}^\top \mathbf{Y} - I \right\|_F^2, \quad (35)$$

where I is the identity matrix corresponding to the dimension of $\mathbf{Y}^\top\mathbf{Y}$.

Once the parameter of the fusion module have been adequately warmed up through preliminary training, the generation loss is introduced to empower the model's capacity to generate accurate answers.

Generation loss. The generation loss ensures the language model produces contextually appropriate textual responses using standard cross-entropy loss, a fundamental objective in language modeling.

The loss targets answer tokens $\langle|\text{ans}|\rangle$ and boundary $\langle|\text{eos}|\rangle$ tokens that terminate each answer. Let $\Omega' = \Omega_{\text{ans}} \cup \Omega_{\text{eos}}$ represent the combined index set of these target tokens. For each $i \in \Omega'$, let $\mathbf{x}_i$ denote the output logits from the language model for the i-th token, where $\mathbf{x}_{i,y_i}$ corresponds to the logit for the ground truth token y_i, and $\mathbf{x}_{i,c}$ represents the logit for candidate token c. The cross-entropy generation loss is defined as:

$$\mathcal{L}_{\text{ce}} = -\frac{1}{|\Omega'|} \sum_{i \in \Omega'} \log \frac{\exp(\mathbf{x}_{i,y_i})}{\sum_c \exp(\mathbf{x}_{i,c})}. \tag{36}$$

5. Experiment

5.1 Setup

This section systematically evaluates the proposed model across multiple vision language benchmarks to investigate the relationship between the number of vision tokens and performance, providing empirical support for the scaling behavior established in Section 3.

Training procedure. The training methodology follows a two-stage approach designed to isolate the effects of vision token scaling. Firstly, a base model is trained with configuration $n_l = 256$ and $n_s = 8$ on the complete dataset to establish a strong foundation. Secondly, this base model is further fine-tuned using 10% subsets of the training data with alternative configurations of n_l and n_s to systematically investigate scaling behavior while maintaining computational efficiency.

Configuration space. To systematically investigate the scaling relationship, the following configurations are considered: 768(8), 512(8), 384(8), 256(8), 128(8), 64(8), 32(8), 16(8), 8(8), and 1(1), where the first number represents n_l (number of learnable queries) and the number in parentheses denotes n_s (number of vision tokens for later questions). This configuration space enables comprehensive analysis of vision token scaling across multiple orders of magnitude.

Evaluation framework. The VLMEvalKit (Duan et al., 2024) is employed to ensure the standardization and reproducibility of evaluations across multiple vision language benchmarks, including MME (Fu et al., 2025), HallusionBench (Guan et al., 2024), POPE (Li et al., 2023c), and other benchmarks.

For detailed description of the datasets employed in the experiments, please refer to Appendix B. Complete implementation details[2], including hyperparameters, training configurations, and computational resources, are provided in Appendix C.

---

[2] The code is available at `https://github.com/tenghuilee/ScalingCapFusedVisionLM.git`

5.2 Analysis of Scaling

This section investigates the scaling properties of model performance with respect to the number of vision tokens n_l. Following the approach proposed in (Kaplan et al., 2020) and Equation (24), and consider the simplified constant setting, $\alpha(n) \to \alpha$, the relationship between the performance metric $S(n_l)$ and the number of vision tokens n_l can be expressed as:

$$S(n_l) \approx \frac{c}{n_l^{\alpha}}. \tag{37}$$

The parameter c reflects baseline model performance, while α determines the rate at which performance changes as a function of token count. Smaller α values indicate slower performance degradation with token reduction, while larger values signify more pronounced impact. The fitted curves provide guidance for optimizing vision language models by balancing computational resources with desired performance levels. Detailed visualizations are provided in Appendix F.

To improve numerical stability, logarithms are applied, and the curves are fitted via the following mean square optimization problem:

$$\underset{\alpha, z}{\operatorname{argmin}} \frac{1}{|\Omega_{n_l}|} \sum_{n \in \Omega_{n_l}} \left(\log(S(n_l)) - (\log(c) - \alpha \log(n_l))\right)^2 , \tag{38}$$

where $|\Omega_{n_l}|$ denotes the number of sampled points. Using the default BFGS optimizer from SciPy, performance data is fitted to generate the curves illustrated in Figures 9 and 10, with corresponding fitted parameters c and α summarized in Table 1.

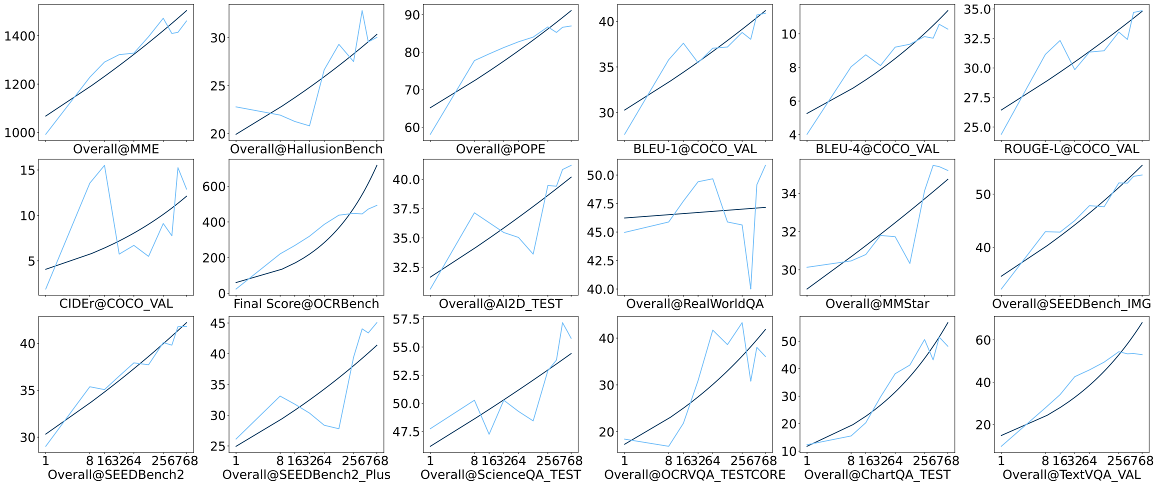


Figure 9: Scaling behavior of performances with respect to n_l. The models are taking vision tokens, user's questions and learnable queries as input ("Vision Question Queries"). The x-axis in each subplot is log scalded, $\log_2(n_l)$. The deep blue lines stand for the fitted curves, and the light blue lines represent the original data points.

The fitted curves reveal a clear logarithmic relationship between the number of vision tokens n_l and performance across most benchmarks, regardless of whether the models

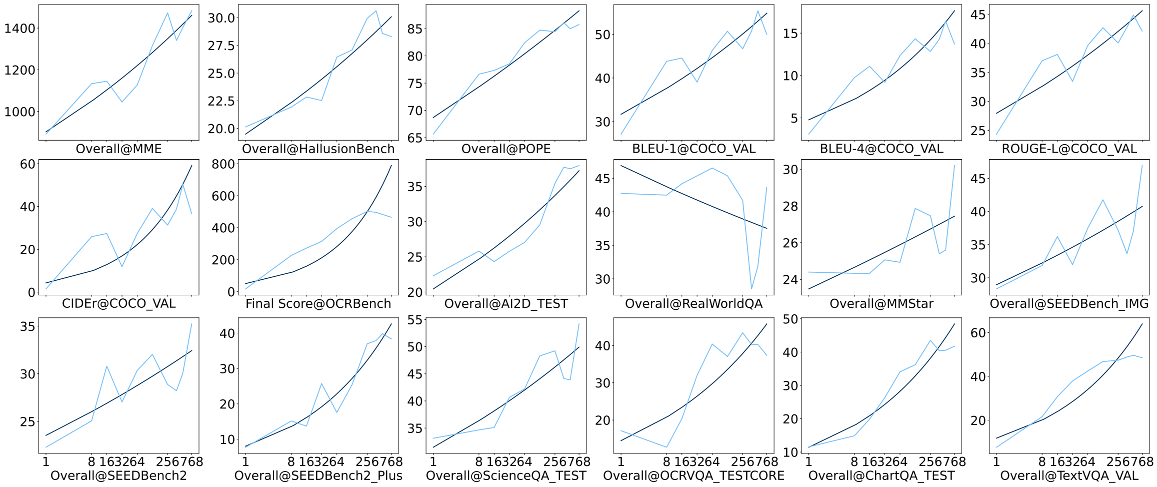


Figure 10: Scaling behavior of performances with respect to n_l. The further fine-tuned models are taking vision tokens and learnable queries as input ("Vision Queries (ft)"). The x-axis in each subplot is log scalded, $\log_2(n_l)$. The deep blue lines stand for the fitted curves, and the light blue lines represent the original data points.

incorporate the user's questions as input. This trend aligns well with the theoretical scaling behavior described in Equation 37. The parameter α represents the slope of the logarithmic relationship, indicating the rate of performance improvement as the number of tokens increases. Across the benchmarks, α values are consistently negative, reflecting diminishing returns in performance gains as the token count grows larger.

In benchmarks such as OCRBench and ChartQA, steeper negative α values highlight that performance is more sensitive to the number of vision tokens. This indicates a stronger dependence on token count for these tasks, where reducing tokens could lead to substantial performance drops. Conversely, benchmarks like ScienceQA TEST and MMStar exhibit relatively flat α values, suggesting that performance is less sensitive to token scaling.

The constant c, derived from the fitted curves, provides an additional perspective on baseline performance. Higher values of c correspond to benchmarks with inherently stronger initial performance, even with fewer tokens. For example, benchmarks such as HallusionBench and ScienceQA TEST exhibit larger c values, indicating robustness in initial model performance.

Through Figures 9 and 10, a significant performance degradation can be observed in the RealWorldQA benchmark at sequence lengths $n_l = 384$ and $n_l = 512$, which results in a failure to fit the theoretical scaling law. Analysis of the evaluation logs reveals that this performance drop stems from incorrect response formatting rather than fundamental model limitations.

For instance, when presented with the question "Is there any flower in the picture?", the expected response format requires selecting option "A: Yes" or "B: No". However, the model generates `<st>flowers<ed>␣<bbox>999,569,1009,609</bbox>␣</st>`, which constitutes an invalid answer format. Similarly, for the question "Which object is closer?", where the correct response should be chosen from "A: The stop sign." or "B: The school zone sign.",

Table 1: The fitted scaling parameters, c, α, for each benchmark as illustrated in Figure 9 and Figure 10. The columns starting with "Vision Question Queries" are for models taking vision tokens, user's questions and learnable queries as input. The columns starting with "Vision Queries (ft)" are for further fine-tuned models taking vision tokens and learnable queries as input.

Benchmark	c	α	c	α
Overall@MME	1.0676e+03	-0.0516	9.0251e+02	-0.0725
Overall@HallusionBench	1.9935e+01	-0.0632	1.9491e+01	-0.0653
Overall@POPE	6.5197e+01	-0.0503	6.8697e+01	-0.0377
BLEU-1@COCO VAL	3.0271e+01	-0.0463	3.1687e+01	-0.0826
BLEU-4@COCO VAL	5.2622e+00	-0.1161	4.7702e+00	-0.1968
ROUGE-L@COCO VAL	2.6441e+01	-0.0413	2.7957e+01	-0.0737
CIDEr@COCO VAL	4.0698e+00	-0.1642	4.3223e+00	-0.3937
Final Score@OCRBench	6.0272e+01	-0.3727	5.0303e+01	-0.4142
Overall@AI2D TEST	3.1651e+01	-0.0359	2.0443e+01	-0.0902
Overall@RealWorldQA	4.6232e+01	-0.0030	4.6893e+01	0.0335
Overall@MMStar	2.8996e+01	-0.0272	2.3483e+01	-0.0235
Overall@SEEDBench IMG	3.4562e+01	-0.0710	2.8942e+01	-0.0516
Overall@SEEDBench2	3.0322e+01	-0.0498	2.3532e+01	-0.0483
Overall@SEEDBench2 Plus	2.4944e+01	-0.0762	8.0294e+00	-0.2512
Overall@ScienceQA TEST	4.6179e+01	-0.0247	3.1462e+01	-0.0695
Overall@OCRVQA TESTCORE	1.7333e+01	-0.1326	1.4473e+01	-0.1734
Overall@ChartQA TEST	1.1750e+01	-0.2370	1.1438e+01	-0.2173
Overall@TextVQA VAL	1.4812e+01	-0.2297	1.1842e+01	-0.2538

the model produces `<st>sign<ed>␣<bbox>899,359,999,419</bbox>␣Answer:␣B.`, which cannot be correctly parsed by the evaluation script despite containing the intended answer.

This formatting issue occurs most frequently in the "Vision Queries (ft)" configuration. The underlying cause likely stems from the fact that the original model was trained with user questions, while this configuration underwent fine-tuning using only 10% of the sampled data without the questions in fusion module. Consequently, the model demonstrates a poor adaptation to evaluation formats that lack conversational context, resulting in responses that fail to conform to the required answer structure.

When these anomalous data points are excluded from the analysis, the fitted curves exhibit substantially improved consistency with the theoretical scaling behavior. An illustration of the corrected scaling relationship, with data points at $n_l = 384$ and $n_l = 512$ removed, is presented in Figure 11.

While these fitted curves are not perfect, they sufficiently capture the overall scaling behavior across different benchmarks. The variations in α and c highlight task-specific differences in sensitivity to token scaling, offering insights for optimizing model architectures.

The fitted curves underscore the scaling capabilities of vision language models and provide a foundation for developing strategies that balance computational efficiency with task-specific performance. The analysis reveals that: the model demonstrates scaling capability following $S(n_l) \approx c/n_l^{\alpha}$, consistent with the theoretical scaling behavior. This behavior is robust

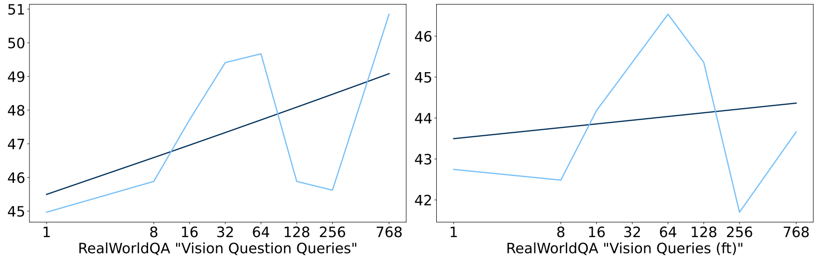


Figure 11: Fitted scaling curves for RealWorldQA benchmark with excluding anomalous data points at $n_l = 384$ and $n_l = 512$ excluded. For RealWorldQA “Vision Question Queries”: $c = 45.496627$, $\alpha = -0.011420$. For RealWorldQA “Vision Queries (ft)”: $c = 43.495766$, $\alpha = -0.002974$. The removal of these outliers demonstrates improved adherence to the theoretical scaling law.

across different input configurations, and does not significantly be effected by the inclusion or exclusion of user questions in the input.

5.3 Analysis on Fusing User Questions

In subsection 3.2, input patterns are formalized into a uniform representation:

$$\underbrace{[\langle|\text{txt}|\rangle \ldots \langle|\text{txt}|\rangle]}_{\text{non-referential}} \overbrace{[\langle|\text{txt}|\rangle \ldots \langle|\text{txt}|\rangle]\,[\langle|\text{vis}|\rangle \ldots]}^{\text{vision-referencing}}.$$

With this representation, text prompt tokens that relate to visual content are classified as vision-referencing components. In practical applications, the input text prompt corresponds to the user question. This formalization enables us to investigate how the presence of vision-referencing information in user questions influences model performance.

The impact of vision-referencing content in user questions can be analyzed from two complementary perspectives.

Enhancing attention mechanisms. On one hand, vision-referencing information enhances attention mechanisms. Vision-referencing content may improve the Vision-Question Fusion model’s ability to understand user intent and focus on relevant image regions. For instance, the question “What is located in the left corner of the image?” contains explicit spatial reference (“left corner”). This spatial reference enabling the fusion model to direct attention to the corresponding image region while disregarding irrelevant areas, potentially yielding more accurate responses.

Pseudo extending vision sequence. On the other hand, vision-referencing tokens increase sequence length. According to the uniform representation framework, if the user’s question contains information related to vision-referencing, it can be considered as a vision-referencing component. This can also be considered as a pseudo expansion of the vision sequence, producing a longer vision token sequence. According to the scaling relationship, the extending in sequence length should correspond to improvement in model performance.

Synthesizing these two mechanisms, it is possible to hypothesize that model performance will improve when user questions contain vision-referencing information, while remaining unaffected when questions lack visual grounding. To empirically validate this hypothesis, an analysis is conducted by leveraging the experimental results from the Subsection 5.2.

Let S_{vqq} denote the performance value of the models that incorporate the user's question as part of the input. For comparison, the model was further fine-tuned without user's question as part of the input, as outlined in Appendix C. Let $S_{\text{vq-ft}}$ represent the performance value of the fine-tuned models that exclude the user's question. The performance difference between the two models, $S_{\text{vqq}} - S_{\text{vq-ft}}$, is visualized as horizontal bar plots in Figure 12. For each horizontal bar plot, the color represents the performance difference $S_{\text{vqq}} - S_{\text{vq-ft}}$. If the value is positive ($S_{\text{vqq}} - S_{\text{vq-ft}} > 0$), the bar is colored green, indicating that the model incorporating the user's question outperforms the fine-tuned model without the question. Conversely, if the value is negative ($S_{\text{vqq}} - S_{\text{vq-ft}} < 0$), the bar is colored orange, indicating that the fine-tuned model without the question achieves better performance than the model with the question.

From Figure 12, it can be observed that green areas (positive performance differences) predominate across most benchmarks, supporting our hypothesis that incorporating user questions with vision-referencing content improves model performance. However, notable exceptions exist where orange areas dominate, indicating that the fine-tuned model without user questions achieves superior performance.

A prominent counterexample is observed in the *COCO VAL* benchmark, which exhibits more orange than green areas. This benchmark employs the question: "Please describe this image in general. Directly provide the description, do not include a prefix like 'This image depicts'.". This question requires comprehensive image descriptions but lacks explicit vision-referencing content such as spatial references or object-specific queries. According to the theoretical framework, such non-referential questions should not provide the attention-focusing benefits described earlier. Instead, they may introduce unnecessary computational overhead without corresponding performance gains, explaining why the fine-tuned model without the question achieves better results in this specific case. This finding aligns with the hypothesis that performance improvements are contingent upon the presence of meaningful vision-referencing information in user questions.

To provide a more direct assessment, the models are evaluated without user questions in their original state, and without additional fine-tuning. This approach isolates the inherent impact of including user questions on model performance.

Let S_{vq} denote the performance of the model without incorporating the user's question as input and without any fine-tuning. The performance difference, $S_{\text{vqq}} - S_{\text{vq}}$, is computed for each benchmark and visualized in Figure 13 with the same color scheme as Figure 12. It can be observed that more green bars are present compared to orange bars, suggesting that the model with the user's question generally outperforms the model without the user's question. Additionally, the relative performance range shown on the right-hand side of Figure 13 spans $[0, 40]$, which is significantly larger than the range observed in Figure 12 ($[0, 17]$). This indicates that in these benchmarks, the model relies heavily on the user's questions to achieve better performance.

Similar to Figure 12, there are some outliers in Figure 13, proposing that under some specific conditions, the model without the user's question can slightly perform better than the model with the user's question.

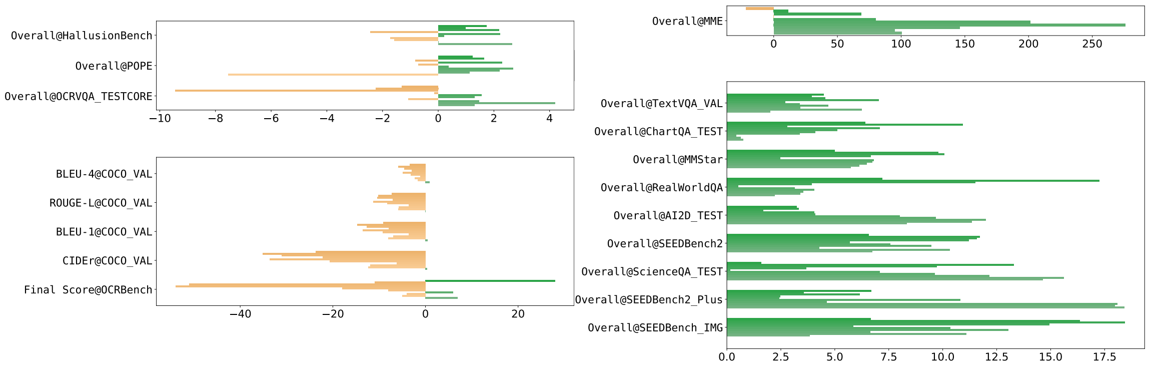


Figure 12: A horizontal bar plot on the performance differences $S_{\text{vqq}} - S_{\text{vq-ft}}$ for different benchmarks, where S_{vqq} is the model with user's questions and $S_{\text{vq-ft}}$ is the further fine-tuned model without user's questions. Each benchmark is represented as a separate plot, and within each plot, the horizontal bars correspond to $n_l = 1, 8, \ldots, 768$, ordered from top to bottom.

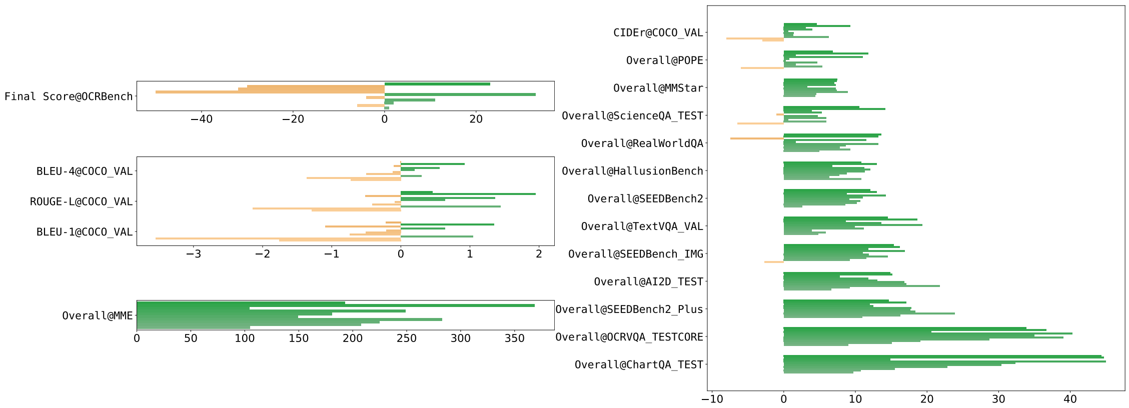


Figure 13: A horizontal bar plot on the performance difference $S_{\text{vqq}} - S_{\text{vq}}$ for each benchmark, where S_{vqq} is the model with user's questions and S_{vq} is the model without user's questions. Each benchmark is represented as a separate plot, and within each plot, the horizontal bars correspond to $N_l = 1, 8, \ldots, 768$, ordered from top to bottom.

This observation suggests that the presence of user questions is possible to improve the model's performance if the user's question is meaningful and relevant to the task. This finding can align with the hypothesis that relevant user questions can provide additional context and guidance to the model, or leading to pseudo extending vision sequences.

6. Conclusion and Future Work

Conclusion. This research establishes a fundamental understanding of how vision token quantity influences model performance in vision-language tasks through both theoretical analysis and empirical validation. The central question of characterizing the relationship between vision tokens and model discriminative capacity has been addressed through a comprehensive mathematical framework that reveals predictable scaling behaviors.

The theoretical analysis demonstrates that the expected divergence between vision-referencing sequences scales according to $\mathbb{E}_{\mathcal{V}}\left[\mathcal{D}(n)\right] = \mathcal{O}\left(\sqrt{\Upsilon(n)}\right)$, exhibiting two distinct scaling regimes: sublinear $\mathcal{O}\left(\sqrt{n}\right)$ scaling for small token counts and linear $\mathcal{O}\left(n\right)$ scaling for large token counts. This theoretical characterization is aligned with the model performance scaling relationships of the form $S(n) \approx c/n^{\alpha(n)}$, where the scaling exponent $\alpha(n)$ depends on the balance between these regimes. Empirical validation across multiple vision-language benchmarks confirms that model performance closely follows these predicted scaling relationships, demonstrating the practical relevance of the theoretical framework.

These findings contribute to understanding vision token scaling through a mathematical framework that reveals predictable performance relationships. The theoretical analysis may provide insights for vision token optimization and architectural design decisions in vision-language systems, complementing existing empirical approaches to multimodal model development.

The research acknowledges certain limitations in its current scope. The theoretical analysis may not hold universally across all model architectures or datasets. Additionally, the empirical validation focuses on a particular model architecture and benchmark suite, which may limit the generalizability of the specific scaling parameters observed.

Future work. Several promising directions emerge from this foundation for future investigation. The theoretical framework can be extended to accommodate more general situations and correlation structures, potentially revealing scaling behaviors in broader classes of vision-language architectures. Investigation of how different vision encoder designs, attention mechanisms, and training procedures influence the scaling parameters $\psi_{\text{equal}}^{(AB)}(n)$, $\psi_{\text{cross}}^{(AA)}(n)$, and $\psi_{\text{cross}}^{(AB)}(n)$ would provide deeper insights into the sources of scaling behavior variations. Broader empirical validation across different model families and architectural approaches would help establish the generalizability of the scaling relationships beyond the configurations examined in this study. Extending the analysis to other modalities, such as video or audio tokens, could reveal whether similar scaling principles apply to multimodal integration more generally.

The finding also suggests practical applications worth exploring, including adaptive vision token allocation strategies and analysis of vision token compression techniques. Wish that the mathematical framework can help to understand these approaches and their impact on model capabilities.

Acknowledgments

The authors would like to express their sincere gratitude to the anonymous reviewers and action editor for their insightful comments and valuable suggestions, which significantly improved the quality of this paper. We are particularly grateful to Dr. Andong Wang for his helpful discussions on the writing.

This study was supported in part by Moonshot R&D Grant Number JPMJMS2239. Tenghui Li was supported by RIKEN's IPA Program.

Appendix A. Additional Discussion for Divergence Token Analysis

A.1 Additional Derivative for Divergence Scaling

The complete derivative for Equation (12), Inequality (13) and (16):

$$
\begin{aligned}
&\mathbb{E}_{\mathcal{V}}\left[\left\|\sum_{i=1}^{n}\left(\mathbf{h}_i^{(A)}-\mathbf{h}_i^{(B)}\right)\right\|_F^2\right]\\
&=\sum_{i=1}^{n}\mathbb{E}_{\mathcal{V}}\left[\left\|\mathbf{h}_i^{(A)}-\mathbf{h}_i^{(B)}\right\|_F^2\right]+\sum_{i,j=1,i\neq j}^{n}\mathbb{E}_{\mathcal{V}}\left[\left(\mathbf{h}_i^{(A)}-\mathbf{h}_i^{(B)}\right)^{\top}\left(\mathbf{h}_j^{(A)}-\mathbf{h}_j^{(B)}\right)\right]\\
&=\sum_{i=1}^{n}\left(\mathbb{E}_{\mathcal{V}}\left[\left\|\mathbf{h}_i^{(A)}\right\|_F^2\right]+\mathbb{E}_{\mathcal{V}}\left[\left\|\mathbf{h}_i^{(B)}\right\|_F^2\right]-2\mathbb{E}_{\mathcal{V}}\left[\left(\mathbf{h}_i^{(A)}\right)^{\top}\mathbf{h}_i^{(B)}\right]\right)\\
&\quad+\sum_{i,j=1,i\neq j}^{n}\left(\mathbb{E}_{\mathcal{V}}\left[\left(\mathbf{h}_i^{(A)}\right)^{\top}\mathbf{h}_j^{(A)}\right]+\mathbb{E}_{\mathcal{V}}\left[\left(\mathbf{h}_i^{(B)}\right)^{\top}\mathbf{h}_j^{(B)}\right]\right)\\
&\quad-\sum_{i,j=1,i\neq j}^{n}\left(\mathbb{E}_{\mathcal{V}}\left[\left(\mathbf{h}_i^{(A)}\right)^{\top}\mathbf{h}_j^{(B)}\right]+\mathbb{E}_{\mathcal{V}}\left[\left(\mathbf{h}_i^{(B)}\right)^{\top}\mathbf{h}_j^{(A)}\right]\right)\\
&=\sum_{i=1}^{n}\left(\mathbb{E}_{\mathcal{V}}\left[\left\|\mathbf{h}_i^{(A)}\right\|_F^2\right]+\mathbb{E}_{\mathcal{V}}\left[\left\|\mathbf{h}_i^{(B)}\right\|_F^2\right]-2\mathbb{E}_{\mathcal{V}}\left[\left\|\mathbf{h}_i^{(A)}\right\|_F\left\|\mathbf{h}_i^{(B)}\right\|_F\cos\left(\Theta_{ii}^{(AB)}\right)\right]\right)\\
&\quad+\sum_{i,j=1,i\neq j}^{n}\left(\mathbb{E}_{\mathcal{V}}\left[\left\|\mathbf{h}_i^{(A)}\right\|_F\left\|\mathbf{h}_j^{(A)}\right\|_F\cos\left(\Theta_{ij}^{(AA)}\right)\right]+\mathbb{E}_{\mathcal{V}}\left[\left\|\mathbf{h}_i^{(B)}\right\|_F\left\|\mathbf{h}_j^{(B)}\right\|_F\cos\left(\Theta_{ij}^{(BB)}\right)\right]\right)\\
&\quad-\sum_{i,j=1,i\neq j}^{n}\left(\mathbb{E}_{\mathcal{V}}\left[\left\|\mathbf{h}_i^{(A)}\right\|_F\left\|\mathbf{h}_j^{(B)}\right\|_F\cos\left(\Theta_{ij}^{(AB)}\right)\right]+\mathbb{E}_{\mathcal{V}}\left[\left\|\mathbf{h}_i^{(B)}\right\|_F\left\|\mathbf{h}_j^{(A)}\right\|_F\cos\left(\Theta_{ij}^{(BA)}\right)\right]\right).\\
&\leq 2M^2\sum_{i=1}^{n}\left(1-2\mathbb{E}_{\mathcal{V}}\left[\cos\left(\Theta_{ii}^{(AB)}\right)\right]\right)\\
&\quad+M^2\sum_{i,j=1,i\neq j}^{n}\left(\mathbb{E}_{\mathcal{V}}\left[\cos\left(\Theta_{ij}^{(AA)}\right)\right]+\mathbb{E}_{\mathcal{V}}\left[\cos\left(\Theta_{ij}^{(BB)}\right)\right]-2\mathbb{E}_{\mathcal{V}}\left[\cos\left(\Theta_{ij}^{(AB)}\right)\right]\right)\\
&\leq 2M^2\sum_{i=1}^{n}\left(1-\psi_{\text{equal}}^{(AB)}(n)\right)+M^2\sum_{i,j=1,i\neq j}^{n}\left(\psi_{\text{cross}}^{(AA)}(n)+\psi_{\text{cross}}^{(BB)}(n)-2\psi_{\text{cross}}^{(AB)}(n)\right)\\
&\leq 2M^2\left(n\left(1-\psi_{\text{equal}}^{(AB)}(n)\right)+(n^2-n)\left(\psi_{\text{cross}}^{(AA)}(n)-\psi_{\text{cross}}^{(AB)}(n)\right)\right)\\
&=\mathcal{O}\left(n\left(1-\psi_{\text{equal}}^{(AB)}(n)\right)+(n^2-n)\left(\psi_{\text{cross}}^{(AA)}(n)-\psi_{\text{cross}}^{(AB)}(n)\right)\right)\\
&:=\mathcal{O}\left(\Upsilon(n)\right).
\end{aligned}
$$

A.2 Analysis of Dependency Measures

This appendix provides a detailed analysis of the constraints imposed by the dependency measures $\psi_{\text{equal}}^{(AB)}$, $\psi_{\text{cross}}^{(AA)}$, and $\psi_{\text{cross}}^{(AB)}$ on the sequence length n and examines the conditions under which meaningful divergence bounds can be established.

Recall the upper bound proposed in Inequality 18, which is given by:

$$\sqrt{n\left(1-\psi_{\text{equal}}^{(AB)}\right)+(n^2-n)\left(\psi_{\text{cross}}^{(AA)}-\psi_{\text{cross}}^{(AB)}\right)}.$$

The fundamental constraint arises from requiring the expression within the square root remain non-negative (in real numbers). Rearranging the fundamental constraint yields:

$$n\left(\psi_{\text{cross}}^{(AA)}-\psi_{\text{cross}}^{(AB)}\right)\geq\psi_{\text{equal}}^{(AB)}+\psi_{\text{cross}}^{(AA)}-\psi_{\text{cross}}^{(AB)}-1, \tag{39}$$

where $n \geq 1$ by definition and all dependency measures are bounded within $[0,1]$. There are three distinct cases based on the relationship between $\psi_{\text{cross}}^{(AA)}$ and $\psi_{\text{cross}}^{(AB)}$.

1. $\psi_{\text{cross}}^{(AA)}=\psi_{\text{cross}}^{(AB)}$: When $\psi_{\text{cross}}^{(AA)}$ and $\psi_{\text{cross}}^{(AB)}$ are equal, the quadratic term in the divergence bound vanishes. In this scenario, constraint (39) is always satisfied for any $n \geq 1$, and the divergence bound is simplified to $\mathcal{O}\left(\sqrt{n}\right)$.

2. $\psi_{\text{cross}}^{(AA)}>\psi_{\text{cross}}^{(AB)}$: This case represents the most comment scenario where temporal similarity within the same branch is greater than that between different branches. The constraint becomes:

$$n\geq\frac{\psi_{\text{equal}}^{(AB)}+\psi_{\text{cross}}^{(AA)}-\psi_{\text{cross}}^{(AB)}-1}{\psi_{\text{cross}}^{(AA)}-\psi_{\text{cross}}^{(AB)}}. \tag{40}$$

Since $\psi_{\text{equal}}^{(AB)}\in[0,1]$ and $\psi_{\text{cross}}^{(AA)}-\psi_{\text{cross}}^{(AB)}\geq 0$, it can be shown that:

$$\frac{\psi_{\text{equal}}^{(AB)}+\psi_{\text{cross}}^{(AA)}-\psi_{\text{cross}}^{(AB)}-1}{\psi_{\text{cross}}^{(AA)}-\psi_{\text{cross}}^{(AB)}}=1+\frac{\psi_{\text{equal}}^{(AB)}-1}{\psi_{\text{cross}}^{(AA)}-\psi_{\text{cross}}^{(AB)}}\leq 1. \tag{41}$$

Therefore, the constraint is automatically satisfied for all $n \geq 1$, ensuring a meaningful bound can be established for any sequence length.

3. $\psi_{\text{cross}}^{(AA)}<\psi_{\text{cross}}^{(AB)}$: This case represents an unusual scenario where the temporal similarity between different branches is greater than that within the same branch. While this is theoretically possible, it is highly unlikely in practical applications because the similarity of different tokens in different branches should not be higher than that in the same branch. Although this case is practically unlikely, it is still valid to analyze here, and the constraint becomes,

$$n\leq\frac{\psi_{\text{equal}}^{(AB)}+\psi_{\text{cross}}^{(AA)}-\psi_{\text{cross}}^{(AB)}-1}{\psi_{\text{cross}}^{(AA)}-\psi_{\text{cross}}^{(AB)}}=1+\frac{\psi_{\text{equal}}^{(AB)}-1}{\psi_{\text{cross}}^{(AA)}-\psi_{\text{cross}}^{(AB)}}. \tag{42}$$

Since $\psi_{\text{equal}}^{(AB)}-1\leq 0$ and $\psi_{\text{cross}}^{(AA)}-\psi_{\text{cross}}^{(AB)}<0$, the fraction $\frac{\psi_{\text{equal}}^{(AB)}-1}{\psi_{\text{cross}}^{(AA)}-\psi_{\text{cross}}^{(AB)}}\geq 0$. It shows that the sequence length n should be less than a threshold. As $\psi_{\text{cross}}^{(AA)}-\psi_{\text{cross}}^{(AB)}$ approaches zero, this fraction can become arbitrarily large, leading to:

$$0\leq\frac{\psi_{\text{equal}}^{(AB)}-1}{\psi_{\text{cross}}^{(AA)}-\psi_{\text{cross}}^{(AB)}}\leq+\infty. \tag{43}$$

This imposes an upper bound on the sequence length n for which meaningful divergence bounds can be established. If the actual sequence length exceeds this bound, the theoretical framework cannot guarantee sequence separability, regardless of the length of the context. However, this scenario is considered unlikely in practical applications, as it would require cross-branch correlations to systematically exceed intra-sequence temporal correlations.

To provide a direct empirical validation of the theoretical relationships, the estimated values of $\psi^{(AA)}_{\text{cross}}(n)$ and $\psi^{(AB)}_{\text{cross}}(n)$ are compared in Figure 14. The experimental configuration follows the same settings established in Subsection 3.6, utilizing the identical results. As demonstrated in Figure 14, the empirically estimated $\psi^{(AA)}_{\text{cross}}(n)$ consistently exceeds $\psi^{(AB)}_{\text{cross}}(n)$ across all experimental conditions, thereby confirming the validity of the theoretical framework presented above.

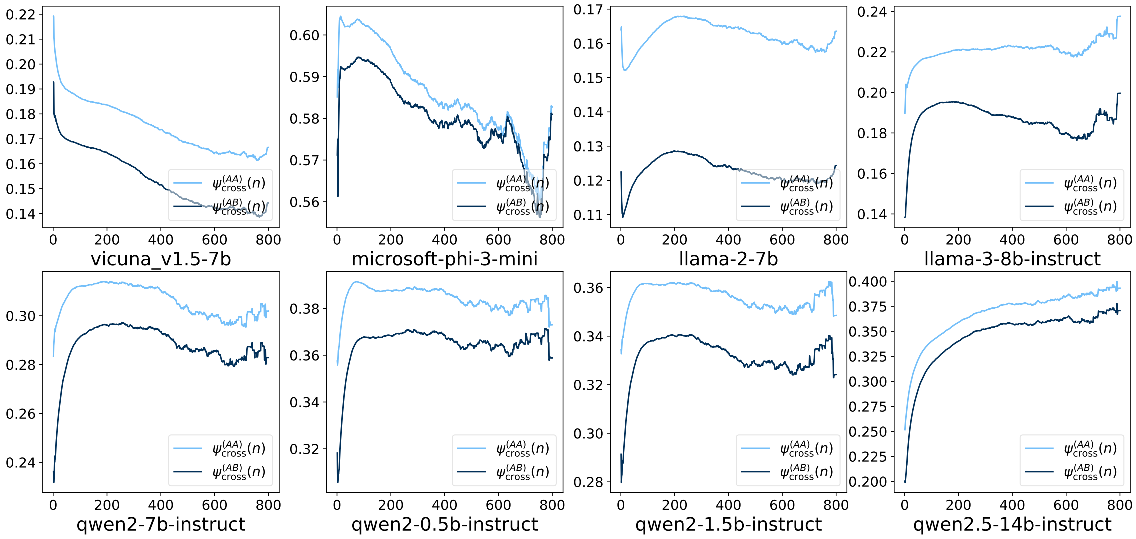


Figure 14: Estimated $\psi^{(AA)}_{\text{cross}}(n)$ and $\psi^{AB}_{\text{cross}}(n)$ for various large language models.

A.3 Expression for Scaling Exponent

This appendix provides additional analysis for the scaling relationship $\gamma\Upsilon(n)^{\beta/2} = c/n^{\alpha(n)}$, and express a closed-form expression for scaling constant c and scaling exponent $\alpha(n)$. γ is the scaling factor for matching $S(n)$ and $\Upsilon(n)^{\beta/2}$, and $\beta > 0$ measures the exponent relationship between $\sqrt{\Upsilon(n)}$ and $c/n^{\alpha(n)}$.

A special situation arises when $n = 1$, where the quadratic component in $\Upsilon(n)$ vanishes, therefore, without loss of generality, it can be considered that

$$c = \gamma\Upsilon(1)^{\beta/2} = \gamma\left(1 - \psi^{(AB)}_{\text{equal}}(1)\right)^{\frac{\beta}{2}}. \tag{44}$$

1. If $\psi_{\text{cross}}^{(AA)}(n) - \psi_{\text{cross}}^{(AB)}(n) = 0$ for all $n > 1$, the scaling exponent is derived as follows:

$$\begin{aligned}\frac{c}{n^{\alpha(n)}} &= \gamma\left(n\left(1-\psi_{\text{equal}}^{(AB)}(n)\right)\right)^{\frac{\beta}{2}},\\ \log\frac{c}{\gamma} - \alpha(n)\log n &= \frac{\beta}{2}\left(\log n + \log\left(1-\psi_{\text{equal}}^{(AB)}(n)\right)\right).\end{aligned}$$

The solution is therefore given by

$$\alpha(n) = \log_n\frac{c}{\gamma} - \frac{\beta}{2} - \frac{\beta}{2}\log_n\left(1-\psi_{\text{equal}}^{(AB)}(n)\right). \tag{45}$$

2. If $\psi_{\text{cross}}^{(AA)}(n) - \psi_{\text{cross}}^{(AB)}(n) > 0$ for all $n > 1$, the following relationships hold,

$$\begin{aligned}\frac{c}{n^{\alpha(n)}} &= \gamma\left(n\left(1-\psi_{\text{equal}}^{(AB)}(n)\right) + (n^2-n)\left(\psi_{\text{cross}}^{(AA)}(n)-\psi_{\text{cross}}^{(AB)}(n)\right)\right)^{\frac{\beta}{2}},\\ \log\frac{c}{\gamma} - \alpha(n)\log n &= \frac{\beta}{2}\log n + \frac{\beta}{2}\log\left(1-\psi_{\text{equal}}^{(AB)}(n) + (n-1)\left(\psi_{\text{cross}}^{(AA)}(n)-\psi_{\text{cross}}^{(AB)}(n)\right)\right).\end{aligned}$$

Therefore, $\alpha(n)$ becomes,

$$\alpha(n) = \log_n\frac{c}{\gamma} - \frac{\beta}{2} - \frac{\beta}{2}\log_n\left(1-\psi_{\text{equal}}^{(AB)}(n) + (n-1)\left(\psi_{\text{cross}}^{(AA)}(n)-\psi_{\text{cross}}^{(AB)}(n)\right)\right).$$

The logarithmic term in the expression can be reformulated by factoring out the quadratic coefficient,

$$\begin{aligned}&\log_n\left(1-\psi_{\text{equal}}^{(AB)}(n) + (n-1)\left(\psi_{\text{cross}}^{(AA)}(n)-\psi_{\text{cross}}^{(AB)}(n)\right)\right)\\ &= \log_n\left(\psi_{\text{cross}}^{(AA)}(n)-\psi_{\text{cross}}^{(AB)}(n)\right) + \log_n\left(n + \frac{1-\psi_{\text{equal}}^{(AB)}(n)}{\psi_{\text{cross}}^{(AA)}(n)-\psi_{\text{cross}}^{(AB)}(n)} - 1\right)\\ &= \log_n\left(\psi_{\text{cross}}^{(AA)}(n)-\psi_{\text{cross}}^{(AB)}(n)\right) + \log_n\left(n + n|_{\rho(n)=1}\right),\end{aligned}$$

where the final equality comes from the definition of the critical balance point in Equation (21) with slightly abused notation. Based on this decomposition, the scaling exponent $\alpha(n)$ can be expressed in a closed form as:

$$\alpha(n) = \log_n\frac{c}{\gamma} - \frac{\beta}{2} - \frac{\beta}{2}\left(\log_n\left(\psi_{\text{cross}}^{(AA)}(n)-\psi_{\text{cross}}^{(AB)}(n)\right) + \log_n\left(n + n|_{\rho(n)=1}\right)\right). \tag{46}$$

To simply the analysis, the same constant setting introduced in Subsection 3.5 is applied: $\psi_{\text{equal}}^{(AB)}(n) \to \psi_{\text{equal}}^{(AB)}$, $\psi_{\text{cross}}^{(AA)}(n) \to \psi_{\text{cross}}^{(AA)}$, and $\psi_{\text{cross}}^{(AB)}(n) \to \psi_{\text{cross}}^{(AB)}$. With this simplification,

$$c = \gamma\left(1-\psi_{\text{equal}}^{(AB)}\right)^{\frac{\beta}{2}},$$

and the scaling exponent $\alpha(n)$ can also be simplified.

For $n > 1$ and $\psi_{\text{cross}}^{(AA)} - \psi_{\text{cross}}^{(AB)} = 0$, the scaling exponent becomes,

$$\begin{aligned}\alpha(n) &= \log_n \frac{c}{\gamma} - \frac{\beta}{2} - \frac{\beta}{2}\log_n\left(1 - \psi_{\text{equal}}^{(AB)}\right) \\ &= \frac{\beta}{2}\log_n\left(1 - \psi_{\text{equal}}^{(AB)}\right) - \frac{\beta}{2} - \frac{\beta}{2}\log_n\left(1 - \psi_{\text{equal}}^{(AB)}\right) \\ &= -\frac{\beta}{2}.\end{aligned} \tag{47}$$

For $n > 1$ and $\psi_{\text{cross}}^{(AA)} - \psi_{\text{cross}}^{(AB)} > 0$, the scaling exponent is given by,

$$\begin{aligned}\alpha(n) &= \log_n \frac{c}{\gamma} - \frac{\beta}{2} - \frac{\beta}{2}\left(\log_n\left(\psi_{\text{cross}}^{(AA)} - \psi_{\text{cross}}^{(AB)}\right) + \log_n\left(n + n|_{\rho(n)=1}\right)\right) \\ &= \frac{\beta}{2}\log_n\left(1 - \psi_{\text{equal}}^{(AB)}\right) - \frac{\beta}{2} - \frac{\beta}{2}\left(\log_n\left(\psi_{\text{cross}}^{(AA)} - \psi_{\text{cross}}^{(AB)}\right) + \log_n\left(n + n|_{\rho(n)=1}\right)\right) \\ &= \frac{\beta}{2}\left(\log_n\left(1 - \psi_{\text{equal}}^{(AB)}\right) - \log_n\left(\psi_{\text{cross}}^{(AA)} - \psi_{\text{cross}}^{(AB)}\right) - \log_n\left(n + n|_{\rho(n)=1}\right) - 1\right) \\ &= \frac{\beta}{2}\left(\log_n\left(1 + n|_{\rho(n)=1}\right) - \log_n\left(n + n|_{\rho(n)=1}\right) - 1\right) \\ &= \frac{\beta}{2}\log_n\left(\frac{1 + n|_{\rho(n)=1}}{n + n|_{\rho(n)=1}}\right) - \frac{\beta}{2}.\end{aligned} \tag{48}$$

This $\alpha(n)$ is a monotonically decreasing function of n, and its value will approach to $-\beta$ as n increase. The gradient of $\alpha(n)$ has the following form,

$$\begin{aligned}\frac{d\alpha(n)}{dn} &= \frac{\beta}{2}\frac{-\frac{1}{n+n|_{\rho(n)=1}}\log n - \frac{1}{n}\log\left(\frac{1+n|_{\rho(n)=1}}{n+n|_{\rho(n)=1}}\right)}{(\log n)^2} \\ &\leq \frac{-\beta}{2(\log n)^2\left(n + n|_{\rho(n)=1}\right)}\log\left(\frac{n\left(1 + n|_{\rho(n)=1}\right)}{n + n|_{\rho(n)=1}}\right) \\ &= \frac{-\beta}{2(\log n)^2\left(n + n|_{\rho(n)=1}\right)}\log\left(\frac{1 + n|_{\rho(n)=1}}{1 + \frac{n|_{\rho(n)=1}}{n}}\right).\end{aligned}$$

Since $n > 1$ and $1 + n|_{\rho(n)=1} > 1 + \frac{n|_{\rho(n)=1}}{n}$, the logarithmic term is positive. Therefore, $\frac{d\alpha(n)}{dn} < 0$ and $\alpha(n)$ is monotonically decreasing. The asymptotic behavior as $n \to \infty$ is characterized by

$$\begin{aligned}\lim_{n\to\infty}\alpha(n) &= -\frac{\beta}{2} + \lim_{n\to\infty}\frac{\beta}{2\log n}\log\left(1 + n|_{\rho(n)=1}\right) - \lim_{n\to\infty}\frac{\beta}{2\log n}\log\left(n + n|_{\rho(n)=1}\right) \\ &= -\frac{\beta}{2} - \lim_{n\to\infty}\frac{\beta}{2}\frac{1}{1 + \frac{n|_{\rho(n)=1}}{n}} \\ &= -\beta.\end{aligned}$$

A.4 Empirical Distribution of Cosine Similarities

To provide empirical validation for the theoretical framework, particularly the dependency assumptions based on cosine similarity distributions, the empirical distributions of cosine similarities are analyzed. The experimental configurations follow those established in Subsection 3.6.

The cosine similarities $\cos(\Theta)_{ii}^{(AB)}$ and $\cos(\Theta)_{ij}^{(AB)}$ are computed, and their distributions are examined through histogram analysis. As discussed in Subsection 3.6, the inherent structure of the dataset results in branch A containing all 'chosen' responses and branch B containing all 'rejected' responses. Consequently, $\cos(\Theta)_{ij}^{(AA)}$ is estimated as the average of cosine similarities within the respective response categories,

$$\cos(\Theta)_{ij}^{(AA)} \leftarrow \frac{1}{2}\left(\cos(\Theta)_{ij}^{(++)} + \cos(\Theta)_{ij}^{(--)}\right),$$

where $\cos(\Theta)_{ij}^{(++)}$ represents cosine similarities between 'chosen' responses and $\cos(\Theta)_{ij}^{(--)}$ represents cosine similarities between 'rejected' responses.

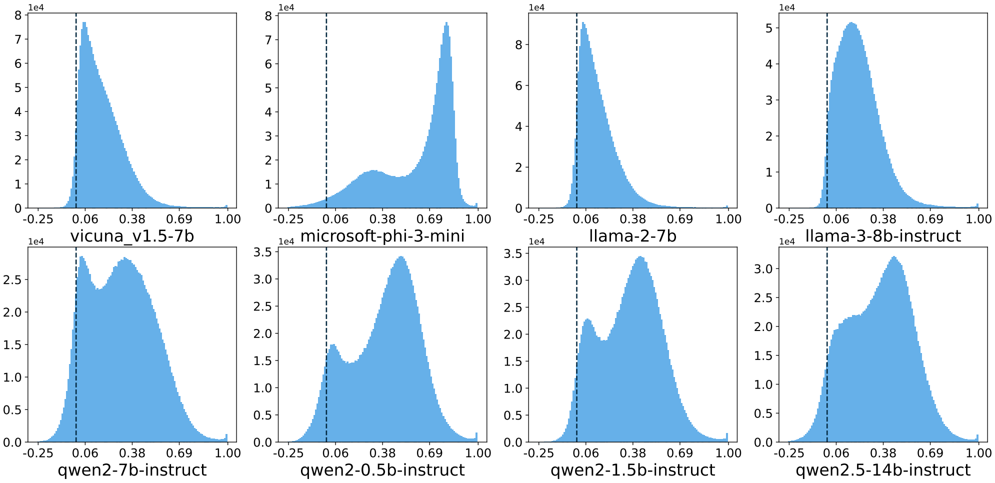
Figure 15 presents the empirical distributions of these cosine similarities. The analysis reveals several key observations: First, the vast majority of cosine similarity values are positive, supporting the theoretical assumption that representations maintain directional coherence. Second, different base models exhibit distinct distributional characteristics, suggesting model-specific dependency patterns. Third, the distributions provide empirical evidence for the validity of the dependency measures defined in Definition 1, as the observed similarity patterns align with the theoretical expectations.

Appendix B. Datasets

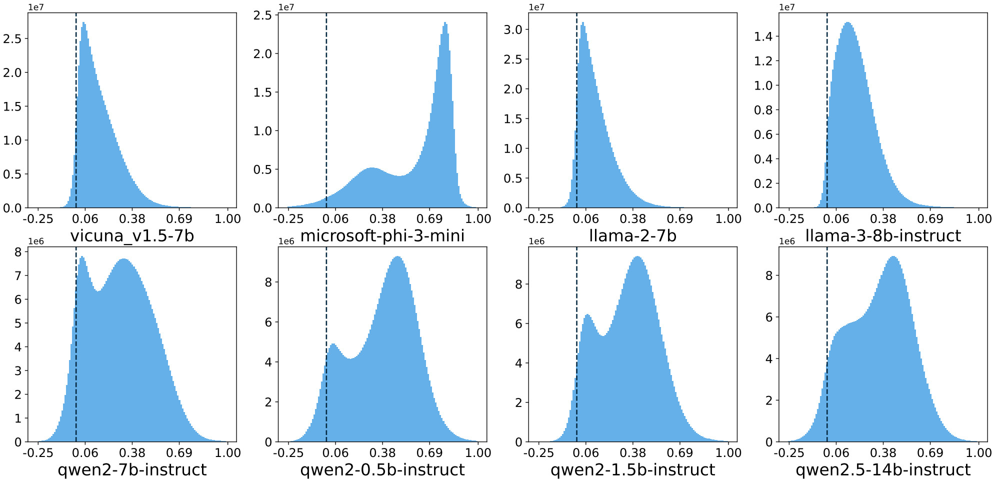
The training utilizes a combination of existing datasets and newly generated data to enhance performance across various tasks. The primary datasets employed include LLAVA V1.5 MIX665K (Liu et al., 2023), BAAI-SVIT (Zhao et al., 2023), and mPLUG DocDownstream 1.0 (Ye et al., 2023).

To further augment the training data, we generated new question-and-answer pairs using detailed descriptions from BAAI-SVIT, with the Llama-3 8B model serving as the agent. The generated questions were of two types: yes/no questions and multiple-choice questions (e.g., selecting one option from A, B, C, or D). To maintain data quality, any generated text that did not conform to the correct format was excluded from the dataset.

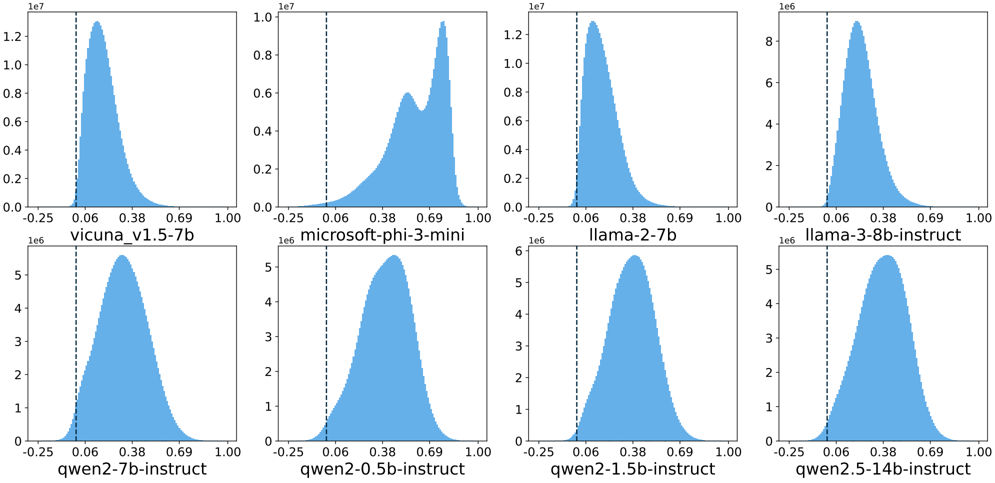
Another source of augmented training data comes from the structure-aware parsing task in CCpdf, derived from mPLUG DocDownstream 1.0. To enhance data quality, we utilized the nougat-base model (Blecher et al., 2024) to convert a subset of images into better-formatted text. To ensure the validity of the outputs, we applied several filtering criteria: (1) generated texts that were too short were discarded, (2) cases where the length difference between the original and generated text was excessively large were excluded, (3) generated texts containing endless repeated patterns were removed, and (4) texts that lacked sufficient similarity to the original under the embedding space, as measured using all-mpnet-base-v2 (Reimers and Gurevych, 2019), were filtered out. These steps ensured that the final dataset maintained high quality and relevance for training.



(a) Histogram of $\cos(\Theta)_{ii}^{(AB)}$



(b) Histogram of $\cos(\Theta)_{ij}^{(AB)}$



(c) Histogram of $\frac{1}{2}\left(\cos(\Theta)_{ij}^{(AA)}+\cos(\Theta)_{ij}^{(BB)}\right)$

Figure 15: Histograms of cosine from Inequality (13). The dashed lines represent $\cos\theta = 0$.

Appendix C. Implementation Details

Our implementation is built upon two widely used open-source pretrained models: CLIP ViT-H/14 (Radford et al., 2021) and Llama-2 7B (Touvron et al., 2023). The CLIP ViT-H/14 vision encoder processes image inputs with a resolution of $(336, 336)$, using a stride of 14, resulting in $24 \times 24 = 576$ tokens, each with a hidden size of 1024. The Llama-2 7B model, serving as the backbone language model, features a hidden size of 4096.

The experiments were conducted on high-performance hardware comprising 8 NVIDIA A100 GPUs, each with 40 GB of memory. For evaluation on more accessible hardware, we utilized NVIDIA RTX A6000 GPUs, each with 48 GB of memory.

The image preprocessing pipeline involves several steps to adapt raw pixel images for input to the CLIP vision encoder. First, as introduced in subsection 4.2, the images are processed into global and local views, with each view independently passed through the CLIP encoder to generate vision tokens.

Based on the available hardware resources, we adopt an HD-9 cropping strategy, which generates nine local image views and one global image view as input to the CLIP vision encoder. Each view has a resolution of $(336, 336)$, resulting in a maximum input image resolution of (1008×1008). This setup initially produces 5760 tokens, which is computationally expensive to process. To address this, we apply a local merging operation that reduces the token count by a factor of four. Neighboring vision tokens are merged in a $4 \times 4 \Rightarrow 1$ manner using a 2D convolution with a kernel size of 3, padding of 1, and a stride of 2. This reduces the number of vision tokens to 1440. After appending $\langle|\text{new line}|\rangle$ tokens to mark the end of each view, the total token count becomes 1450. The tokens for all views are then concatenated, ensuring efficient processing while retaining critical information.

As introduced in subsection 4.2, the fused model integrates the vision tokens from the global and local views with the text tokens from the user's question and the learnable queries. For the first question, n_l learnable queries are appended, while subsequent questions are each followed by n_s learnable queries. The value of n_l ranges from 1 to 768, with n_s set to 8 when $n_l \geq 8$. For smaller values of n_l (i.e., $n_l < 8$), n_s is set equal to n_l. For example, when only one learnable query is used, both n_l and n_s are set to 1. The text tokenizer is based on the CLIP text tokenizer, and n_l new special tokens are added to the tokenizer to support the learnable queries.

The hidden size of the fused model is set to 1024, matching the hidden size of the vision tokens generated by CLIP ViT-H/14. This is significantly smaller than the hidden size of Llama-2 7B, which is 4096. Adopting a smaller hidden size helps reduce both memory consumption and computational cost, making the model more efficient for training and inference.

To align the fused vision tokens generated by the fused model with the Llama-2 7B backbone, a linear projection layer is used to map the fused vision tokens to the hidden size of the Llama-2 model. We consider a three-step training strategy for optimizing the model.

In the first step, we perform preliminary training using the contrastive loss introduced in Equation 35 to align the fused vision tokens with the CLIP text encoder. Since the maximum sequence length of the CLIP text encoder is 77 and most questions exceed this limit, we extend the sequence length to 512 to accommodate longer inputs. The modules involved in this step include the vision encoder, the 2D convolution layer for merging neighboring vision tokens, the fused model, and the CLIP text encoder. This step requires only a few training steps, using a batch size of 32 per device, gradient accumulation of 1, an equivalent batch size of 256, and a learning rate of 2e-5. The training employs a cosine learning rate scheduler with a warm-up ratio of 0.1 and is performed over 1000 steps.

The second step involves fine-tuning on the full dataset. The required modules include the vision encoder, the 2D convolution layer, the fused model, a linear projection layer to map the fused vision tokens to the Llama-2 hidden size, and the Llama-2 7B backbone. In this stage, the Llama-2 backbone is frozen, and the remaining modules are updated. The training uses a batch size of 5 per device, gradient accumulation of 64, and an equivalent batch size of 2560, with a learning rate of 2e-5. A cosine learning rate scheduler with a warm-up ratio of 0.03 is employed. The training spans two epochs and uses the generation loss described in Equation 36. The fusion model consists of 20 layers, with $n_l = 256$ learnable queries for the first question and $n_s = 8$ for subsequent questions. Given the considerable size of the dataset, the fine-tuning process is estimated to require more than 200 hours.

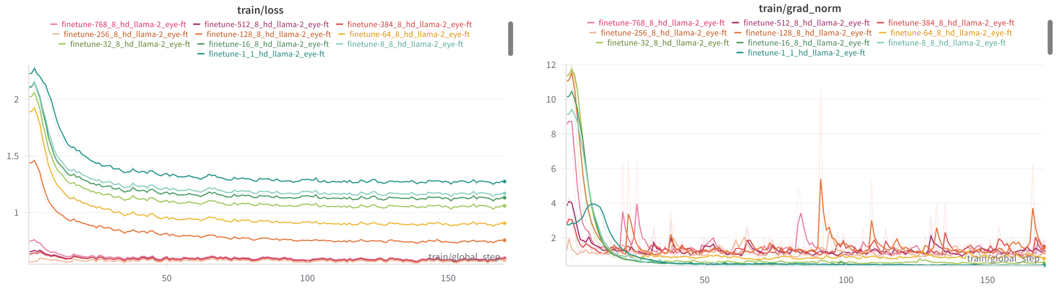
The third step focuses on further fine-tuning to evaluate various configurations of n_l and n_s, such as $n_l = 128, n_s = 8$ and $n_l = 64, n_s = 8$. To reduce the substantial time required for full-data fine-tuning, we reuse the model trained in the second step ($n_l = 256, n_s = 8$) and fine-tune it using only 10% of randomly sampled training data, which will take approximately 10 hours. This includes fine-tuning for alternative configurations as well as the original $n_l = 256, n_s = 8$. This strategy significantly reduces computational cost while allowing for the exploration of multiple settings. The loss function remains the same as in the second step (Equation 36), with the Llama-2 backbone frozen and updates applied only to the other modules. All other hyperparameters and configurations are consistent with those used in the second step.

Another critical aspect of the fusion model is assessing the impact of incorporating the user's question on the model's performance. To evaluate this, we introduce an alternative setup where the model processes inputs without including the user's question. In this configuration, the input is limited to vision tokens and learnable queries, excluding any text-based queries provided by the user. To determine the validity of fusing the user's question, we conduct experiments both with or without further fine-tuning the model, keeping the Llama-2 backbone frozen in both cases. For this evaluation, we reuse the 10% randomly sampled training data previously employed for tuning different configurations of n_l. All other hyperparameters and configurations are kept consistent with those used in the second step. This approach allows us to efficiently analyze the influence of the user's question on the fusion model's performance while maintaining computational efficiency.

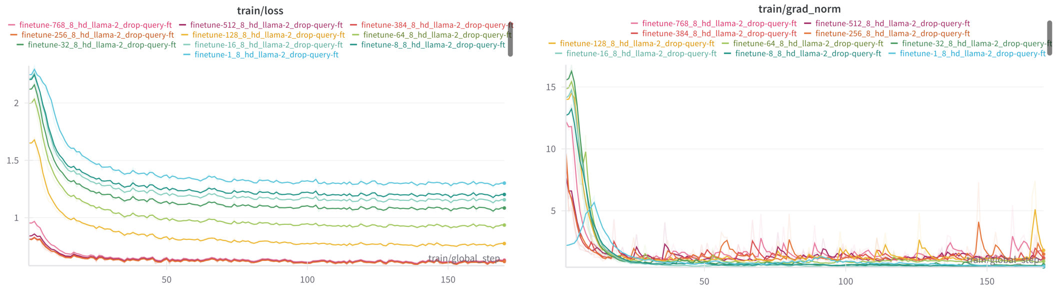
Appendix D. Detailed Results

For evaluation, we utilize the VLMEvalKit (Duan et al., 2024) to compute scores across various benchmarks. Different configurations of n_l and n_s are tested, including: 768(8), 512(8), 384(8), 256(8), 128(8), 64(8), 32(8), 16(8), 8(8), 1(1), where the first number represents n_l and the second number in parentheses denotes n_s. The losses and gradient norms of the further fine-tuned models are shown in Figure 16.

The evaluation spans a diverse set of benchmarks, including MME (Fu et al., 2025), HallusionBench (Guan et al., 2024), POPE (Li et al., 2023c), OCRBench (Liu et al., 2024),



(a) Losses of “Vision Query Queries”. (b) Gradient norms of “Vision Query Queries”.



(c) Losses of “Vision Query (ft)”. (d) Gradient norms of “Vision Query (ft)”.

Figure 16: Loss and gradient norm of further fine-tuned model with different configurations.

COCO VAL (Lin et al., 2014), RealWorldQA[3], MMStar (Chen et al., 2024a), SEEDBench IMG (Li et al., 2024b), SEEDBench2 (Li et al., 2023a), SEEDBench2 Plus (Li et al., 2024a), ScienceQA TEST (Lu et al., 2022), AI2D TEST[4], OCRVQA TESTCORE (Mishra et al., 2019), ChartQA TEST (Masry et al., 2022), and TextVQA VAL (Singh et al., 2019).

The baseline models are: 360VL-70B [5], InstructBLIP-7B (Dai et al., 2023), InternLM-XComposer2-4KHD (Dong et al., 2024), LLaVA-v1-7B, LLaVA-v1.5-13B, LLaVA-v1.5-7B (Liu et al., 2023), MiniGPT-4-v1-7B, MiniGPT-4-v2 (Zhu et al., 2024), mPLUG-Owl2 (Ye et al., 2024), OpenFlamingo v2 (Alayrac et al., 2022), Phi-3-Vision (Abdin et al., 2024), Qwen-VL-Chat (Bai et al., 2023b), Fuyu-8B (Bavishi et al., 2023) (jointly pretrained with vision data), GPT-4o (0513, detail-low) (OpenAI, 2024). Results are summarized in the following tables, with all benchmark data for the models obtained from VLMEvalKit (Duan et al., 2024). The blank cells indicate that the scores are not available from VLMEvalKit.

In the column titled “Fused Model Inputs,” the following terms are used to describe the input configurations of the model:

- “Vision Query Queries” refers to the model that takes vision tokens, text tokens from the user’s question, and learnable queries as inputs.
- “Vision Query (ft)” refers to the model that takes vision tokens and learnable queries as inputs, with the model further fine-tuned without the user’s question.

---

[3] `https://x.ai/blog/grok-1.5v`

[4] `https://allenai.org/data/diagrams`

[5] `https://github.com/360CVGroup/360VL`

Table 2: Score comparison of POPE.

Fused Model Inputs	n_l	Overall	acc	precision	recall
Vision Question Queries	768	86.977	87.033	87.357	86.600
Vision Queries (ft)	768	85.742	85.456	84.085	87.467
Vision Queries	768	80.158	76.500	69.362	94.933
Vision Question Queries	512	86.626	86.444	85.482	87.800
Vision Queries (ft)	512	84.990	84.400	81.891	88.333
Vision Queries	512	74.835	66.978	60.451	98.200
Vision Question Queries	384	85.254	84.433	80.984	90.000
Vision Queries (ft)	384	86.080	86.233	87.048	85.133
Vision Queries	384	83.572	82.178	77.508	90.667
Vision Question Queries	256	86.734	87.233	90.267	83.467
Vision Queries (ft)	256	84.447	83.289	78.975	90.733
Vision Queries	256	75.719	68.767	61.933	97.400
Vision Question Queries	128	84.016	82.878	78.778	90.000
Vision Queries (ft)	128	84.727	84.822	85.261	84.200
Vision Queries	128	83.255	82.522	80.331	86.400
Vision Question Queries	64	82.761	81.489	77.440	88.867
Vision Queries (ft)	64	82.398	80.756	76.029	89.933
Vision Queries	64	82.476	81.756	79.343	85.867
Vision Question Queries	32	81.232	80.300	77.562	85.267
Vision Queries (ft)	32	78.539	74.278	67.377	94.133
Vision Queries	32	76.593	78.933	86.167	68.933
Vision Question Queries	16	79.533	77.200	72.150	88.600
Vision Queries (ft)	16	77.338	72.944	66.533	92.333
Vision Queries	16	77.879	75.800	71.717	85.200
Vision Question Queries	8	77.749	74.167	68.280	90.267
Vision Queries (ft)	8	76.634	71.933	65.734	91.867
Vision Queries	8	72.383	63.422	58.140	95.867
Vision Question Queries	1	58.113	49.689	49.778	69.800
Vision Queries (ft)	1	65.648	52.033	51.134	91.667
Vision Queries	1	64.091	58.678	56.679	73.733
Model Name					
LLaVA-v1.5-13B		88.400	88.600	89.600	87.300
360VL-70B		87.300	88.200	94.700	81.100
InstructBLIP-7B		86.100	86.000	85.700	86.500
LLaVA-v1.5-7B		86.100	87.000	92.100	80.900
GPT-4o (0513, detail-low)		85.000	86.200	93.300	78.100
mPLUG-Owl2		84.600	85.400	89.800	79.900
Mantis-8B-Fuyu		84.000	84.000	84.400	83.500
Phi-3-Vision		83.700	85.600	96.100	74.100
LLaVA-v1-7B		75.900	69.200	62.300	96.900
Qwen-VL-Chat		74.900	67.300	67.800	83.600
MiniGPT-4-v2		60.000	48.400	49.900	75.300
OpenFlamingo v2		52.600	42.400	50.000	55.500
MiniGPT-4-v1-7B		34.600	36.400	58.400	24.600
InternLM-XComposer2-4KHD		2.900	10.100	53.700	1.500

- “Vision Query” refers to the model that takes vision tokens and learnable queries as inputs, but remains the same as the original model, without any additional fine-tuning.

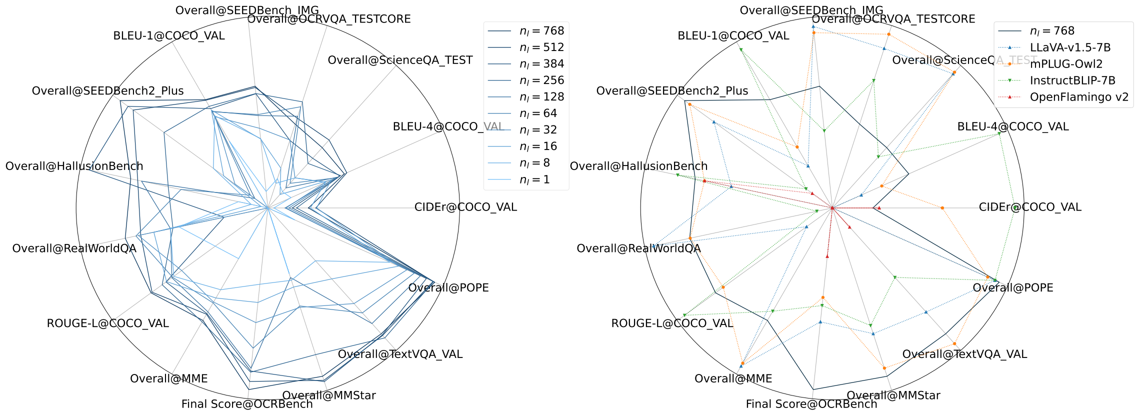
It is important to note that the training datasets for these models differ, and as such, the results are provided for reference purposes only.

Table 3: Score comparison of MME, HallusionBench, COCO VAL.

		MME			HallusionBench				COCO VAL			
Fused Model Inputs	n_l	Overall	Perception	Cognition	aAcc	fAcc	qAcc	Overall	BLEU-1	BLEU-4	ROUGE-L	CIDEr
Vision Question Queries	768	1461.206	1202.635	258.571	50.683	22.254	17.143	30.027	40.892	10.282	34.858	12.897
Vision Queries (ft)	768	1483.059	1244.845	238.214	50.053	18.786	16.044	28.294	49.980	13.713	42.126	36.652
Vision Queries	768	1268.119	1072.405	195.714	34.805	12.428	10.330	19.188	41.113	10.291	34.400	8.308
Vision Question Queries	512	1414.270	1165.699	248.571	50.578	21.387	16.703	29.556	40.707	10.570	34.702	15.256
Vision Queries (ft)	512	1402.772	1182.058	220.714	51.314	17.052	17.363	28.576	55.407	16.420	44.891	50.358
Vision Queries	512	1045.693	860.693	185.000	29.232	10.694	9.890	16.605	39.360	9.647	32.752	6.008
Vision Question Queries	384	1409.437	1152.294	257.143	54.784	21.676	21.978	32.813	38.037	9.740	32.416	7.763
Vision Queries (ft)	384	1340.833	1112.976	227.857	52.156	20.809	18.901	30.622	50.765	14.282	42.812	38.826
Vision Queries	384	1304.939	1085.296	219.643	46.057	15.607	16.484	26.049	39.129	9.844	32.932	4.672
Vision Question Queries	256	1472.529	1209.315	263.214	48.791	18.786	14.945	27.507	38.787	9.838	33.043	9.108
Vision Queries (ft)	256	1472.806	1225.663	247.143	50.473	21.098	18.242	29.938	46.695	12.835	40.083	31.333
Vision Queries	256	1223.633	1018.990	204.643	30.284	10.116	8.352	16.250	38.147	9.280	31.678	5.103
Vision Question Queries	128	1396.146	1157.574	238.571	51.104	18.786	18.022	29.304	37.189	9.383	31.461	5.491
Vision Queries (ft)	128	1315.989	1095.632	220.357	49.527	15.896	15.824	27.082	50.710	14.330	42.714	39.148
Vision Queries	128	1215.086	1001.872	213.214	32.492	10.983	8.132	17.202	37.400	9.181	30.822	4.903
Vision Question Queries	64	1328.092	1095.235	232.857	48.160	16.185	15.604	26.650	37.055	9.207	31.348	6.703
Vision Queries (ft)	64	1126.880	889.023	237.857	48.686	15.029	15.604	26.440	46.324	12.352	39.644	27.363
Vision Queries	64	1178.733	1003.733	175.000	30.599	8.960	6.374	15.311	37.565	9.321	31.435	5.299
Vision Question Queries	32	1321.832	1099.689	222.143	42.587	10.405	9.451	20.814	35.466	8.110	29.854	5.743
Vision Queries (ft)	32	1045.688	830.331	215.357	45.321	12.139	10.110	22.523	39.049	9.191	33.465	11.910
Vision Queries	32	1039.101	835.887	203.214	25.026	6.358	4.615	12.000	36.205	8.608	30.269	4.443
Vision Question Queries	16	1291.082	1066.082	225.000	43.113	13.006	7.692	21.270	37.606	8.754	32.330	15.509
Vision Queries (ft)	16	1145.157	938.729	206.429	44.585	12.717	11.209	22.837	44.580	11.090	38.103	27.403
Vision Queries	16	1066.146	889.003	177.143	27.340	8.382	4.835	13.519	36.562	8.452	30.886	9.226
Vision Question Queries	8	1228.312	992.954	235.357	42.482	13.873	9.451	21.935	35.793	8.040	31.152	13.552
Vision Queries (ft)	8	1133.447	905.590	227.857	42.376	13.295	10.110	21.927	43.824	9.748	37.006	25.907
Vision Queries	8	1020.312	841.383	178.929	34.490	5.491	6.813	15.598	39.338	9.401	33.294	21.574
Vision Question Queries	1	992.141	734.284	257.857	44.059	11.561	12.747	22.789	27.612	4.022	24.384	1.899
Vision Queries (ft)	1	891.690	679.190	212.500	41.956	10.116	8.352	20.141	27.142	3.086	24.364	1.516
Vision Queries	1	887.341	690.556	196.786	25.342	5.780	4.835	11.986	29.373	4.749	25.674	4.887
Model Name												
360VL-70B		2009.700	1646.200	363.600	54.600	26.900	23.100	34.800	71.000	28.100	53.000	86.600
GPT-4o (0513, detail-low)		2328.700	1609.400	719.300	67.500	46.500	41.100	51.700				
InternLM-XComposer2-4KHD		2130.400	1581.500	548.900	60.700	33.800	33.000	42.500	20.200	4.900	17.700	6.000
LLaVA-v1.5-7B		1808.400	1506.200	302.100	48.800	20.500	13.600	27.600	19.800	4.700	20.000	0.000
LLaVA-v1.5-13B		1780.800	1502.600	278.200	45.300	17.100	11.000	24.500	20.700	5.100	21.400	0.400
Qwen-VL-Chat		1860.000	1467.800	392.100	56.400	27.700	26.400	36.800	75.800	34.000	54.900	98.900
mPLUG-Owl2		1786.400	1436.000	350.400	48.900	22.500	16.700	29.400	25.800	7.100	33.600	35.000
Phi-3-Vision		1508.000	1205.100	302.900	56.800	29.500	30.800	39.000	15.800	2.900	16.100	0.000
InstructBLIP-7B		1391.400	1137.100	254.300	53.600	20.200	19.800	31.200	56.800	20.900	39.900	58.100
Mantis-8B-Fuyu		1321.600	1057.700	263.900	50.800	20.800	17.800	29.800	18.500	3.700	18.200	0.000
LLaVA-v1-7B		1075.500	832.000	243.600	43.600	13.000	8.100	21.600	27.000	6.700	26.400	5.500
MiniGPT-4-v1-7B		1047.400	770.600	276.800	52.400	17.300	25.900	31.900	19.600	4.300	17.500	0.800
MiniGPT-4-v2		968.400	708.400	260.000	52.600	16.500	21.100	30.000	12.600	1.400	13.300	0.100
OpenFlamingo v2		607.200	535.000	72.100	52.700	17.600	18.000	29.400	6.400	1.300	15.800	14.900

Table 4: Score comparison of OCRBench, AI2D TEST, RealWorldQA, MMStar, SEEDBench IMG, SEEDBench2, SEEDBench2 Plus, ScienceQA TEST, OCRVQA TESTCORE, ChartQA TEST, TextVQA VAL.

Fused Model Inputs	n_l	Final Score OCRBench	Overall AI2D TEST	Overall RealWorldQA	Overall MMStar	Overall SEEDBench IMG	Overall SEEDBench2	Overall SEEDBench2 Plus	Overall ScienceQA TEST	Overall OCRVQA TESTCORE	Overall ChartQA TEST	Overall TextVQA VAL
Vision Question Queries	768	493.000	41.192	50.850	35.200	53.598	41.804	45.059	55.776	36.068	48.160	52.998
Vision Queries (ft)	768	465.000	37.953	43.660	30.200	46.944	35.230	38.384	54.189	37.370	41.760	48.522
Vision Queries	768	470.000	26.295	37.255	27.733	38.245	29.761	30.391	45.216	2.181	3.800	38.474
Vision Question Queries	512	472.000	40.835	49.150	35.400	53.338	41.775	43.390	57.164	38.021	51.480	53.550
Vision Queries (ft)	512	483.000	37.500	31.895	25.600	37.001	30.064	39.833	43.877	40.267	40.560	49.612
Vision Queries	512	502.000	25.712	35.948	28.000	37.149	28.825	26.263	42.985	1.367	6.800	34.884
Vision Question Queries	384	445.000	39.411	40.000	35.467	52.066	39.806	44.049	53.842	30.794	43.200	53.400
Vision Queries (ft)	384	496.000	37.727	28.497	25.400	33.635	28.222	37.901	44.125	40.234	40.400	48.856
Vision Queries	384	477.000	31.574	47.451	28.400	40.289	31.004	32.060	49.926	10.189	28.320	44.720
Vision Question Queries	256	448.000	39.475	45.621	34.133	52.108	40.072	39.438	52.900	43.294	50.600	54.396
Vision Queries (ft)	256	502.000	35.427	41.699	27.467	37.170	28.874	36.978	49.232	43.424	43.520	47.370
Vision Queries	256	498.000	27.655	34.118	26.867	35.216	25.838	26.921	47.595	3.027	5.600	40.820
Vision Question Queries	128	438.000	33.614	45.882	30.333	47.625	37.713	27.800	48.438	38.607	41.320	49.472
Vision Queries (ft)	128	456.000	29.534	45.359	27.867	41.779	32.034	25.384	48.290	37.044	36.200	46.774
Vision Queries	128	405.000	20.596	44.183	27.067	36.615	26.683	9.969	49.430	3.646	9.000	30.138
Vision Question Queries	64	386.000	35.039	49.673	31.733	47.829	37.914	28.371	49.281	41.699	38.160	45.802
Vision Queries (ft)	64	394.000	27.040	46.536	24.933	37.472	30.343	17.567	42.191	40.397	34.080	42.438
Vision Queries	64	390.000	18.199	36.471	24.467	35.954	28.797	10.716	44.571	2.637	7.800	35.918
Vision Question Queries	32	320.000	35.460	49.412	31.800	45.025	36.507	30.347	50.273	30.924	29.680	42.600
Vision Queries (ft)	32	314.000	25.777	45.359	25.067	31.984	27.032	25.736	40.654	31.999	26.320	37.910
Vision Queries	32	309.000	18.329	40.784	24.400	30.516	25.830	11.989	44.373	2.214	6.840	31.434
Vision Question Queries	16	268.000	36.302	47.712	30.800	42.854	35.066	31.796	47.248	21.810	20.320	34.112
Vision Queries (ft)	16	272.000	24.320	44.183	24.333	36.207	30.795	13.702	35.102	20.345	19.880	30.708
Vision Queries	16	266.000	14.508	39.869	21.867	31.366	24.890	7.905	46.653	2.734	4.840	30.236
Vision Question Queries	8	221.000	37.144	45.882	30.467	42.938	35.370	33.114	50.273	16.895	15.520	27.886
Vision Queries (ft)	8	226.000	25.810	42.484	24.333	31.844	25.058	15.152	34.655	12.695	14.880	21.636
Vision Queries	8	227.000	27.947	36.601	25.933	33.699	26.765	16.864	44.323	1.823	4.760	22.032
Vision Question Queries	1	24.000	30.635	44.967	30.133	32.153	29.022	26.131	47.744	18.424	12.320	9.708
Vision Queries (ft)	1	17.000	22.312	42.745	24.400	28.309	22.289	7.729	33.118	17.122	11.560	7.694
Vision Queries	1	23.000	24.028	40.000	25.733	34.865	26.474	15.195	54.239	9.440	2.640	4.930
Model Name												
InternLM-XComposer2-4KHD		675.000		63.300	55.300	74.600	57.400	65.200	96.300			
GPT-4o (0513, detail-low)		663.000		68.600	61.600	76.400		64.000	90.100			
Phi-3-Vision		637.000		58.800	47.700	70.900	55.700	64.200	90.000	61.900	81.800	72.400
Qwen-VL-Chat		488.000		49.300	34.500	64.800		46.000	68.800	58.600	49.800	60.700
360VL-70B		397.000		62.400	48.100	73.100		52.300	87.400			
Mantis-8B-Fuyu		366.000		43.700	34.400	59.300		33.600	56.800	54.200	53.300	49.000
LLaVA-v1.5-13B		337.000		55.300	34.300	68.200		44.600	72.600	63.400	18.200	48.900
LLaVA-v1.5-7B		318.000		54.800	33.100	65.800	43.400	41.300	69.200	60.600	17.800	45.500
InstructBLIP-7B		276.000		36.900	32.700	44.500		29.500	54.100	50.200	10.900	33.600
LLaVA-v1-7B		269.000		45.800	27.100	50.400		31.200	61.800			
mPLUG-Owl2		255.000		50.800	34.800	64.500		44.400	69.500	65.200	22.800	56.400
MiniGPT-4-v1-7B		172.000		21.300	16.300	31.600		15.200	39.600	0.300		0.600
OpenFlamingo v2		149.000		35.200	26.900	28.800		28.700	44.800	8.900		16.300
MiniGPT-4-v2		31.000		30.700	21.300	29.400		23.300	54.700	0.000		0.000



(a) Performance variation across n_l (b) Comparison of $n_l = 768$ with baseline

Figure 17: Performance comparison between the proposed model and baseline models across various benchmarks. For normalization purposes, each benchmark's performance scores are scaled to the interval $[0, 1]$ using the transformation $\mathbf{x} \leftarrow (\mathbf{x} - \min(\mathbf{x}))/(\max(\mathbf{x}) - \min(\mathbf{x}))$. Baseline model results are obtained from VLMEvalKit evaluations.

Appendix E. Comparison with Other Models And Computation Costs

E.1 Comparison with Benchmark Models

To evaluate the effectiveness of the proposed approach, comprehensive comparisons are conducted against established baseline models across multiple benchmarks. The baseline models considered for comparison include LLaVA-v1.5-7B (Liu et al., 2023), mPLUG-Owl2 (Ye et al., 2024), InstructBLIP-7B (Dai et al., 2023), and OpenFlamingo v2 (Alayrac et al., 2022).

It should be noted that these baseline models were trained on different datasets than the proposed model, rendering the comparisons indicative rather than directly equivalent. The performance evaluations are therefore presented for reference and contextual understanding of the model's capabilities.

Figure 17a shows the performance variation of the proposed model with different values of n_l. The light blue color corresponds to the proposed model with smaller n_l, and the dark blue color corresponds to the proposed model with larger n_l. The radar chart shows that increasing n_l improves the model's performance across most benchmarks, consisting with the theoretical analysis presented earlier.

The comparative analysis, as illustrated in Figure 17b, reveals the performance characteristics of both the proposed model (represented by solid line) and the baseline models (represented in dashed lines). The proposed model demonstrates competitive performance relative to baseline models on several evaluation benchmarks, including OCRBench, MMStart POPE, SEEDBench2 Plus, and HallusionBench. These results suggest that the model effectively captures the underlying patterns required for these specific tasks. However, performance gaps are observed on other benchmarks, including ScienceQA Test, SeedBench IMG, RealWorld QA, and MME, where the proposed model underperforms compared to established baselines. These disparities may be attributed to differences in training data, model architecture design choices, or task-specific optimization strategies employed by the baseline models.

E.2 Comparison with Token Reduction Methods

The experimental comparison evaluates the proposed method against two established token reduction approaches: VisPruner (Zhang et al., 2024), which utilizes visual cues from the vision encoder for token pruning, and VisionZip (Yang et al., 2025), which selects a set of informative tokens for the language model. A comprehensive comparison across multiple benchmarks is conducted to assess the performance of the proposed method relative to these baseline methods. Results are presented in Table 5.

The experimental results reveal that each method exhibits distinct strengths and weaknesses across different task categories. The proposed method shows relatively lower performance on certain benchmarks such as MME and RealWorldQA, where visual perception capabilities are primarily evaluated. However, the proposed method demonstrates better performance on text-intensive tasks, including OCRBench, ChartQA, and TextVQA, suggesting more effective preservation of textual information during token reduction.

E.3 Computational Cost Analysis

The main computational overhead arises from high-dimensional processing and additional fusion modules, which are computed exclusively during the prefill phase. Therefore, it is better to measure the time to first token (TTFT). To address the trade-off between token count and computational efficiency, TTFT and GPU resource utilization are measured.

The evaluations are conducted on 500 randomly selected images from the COCO test set (Lin et al., 2014) using the prompt “Please describe the given image into detail.” . The results in Table 6 demonstrate a clear computational trade-off pattern. TTFT decreases substantially from approximately 0.3 seconds at larger n_l to around 0.2 seconds at smaller n_l. GPU utilization follows a similar trend, dropping from over 80% to approximately 73-77%. Memory usage remains relatively stable across all configurations. It is notable that introducing two additional modules, the high-dimensional processing modules and fusion modules, will incur additional computational costs, therefore more computational resources are required than the methods without these modules.

To further quantify the computational trade-offs associated with varying vision token counts, the number of floating-point operations (FLOPs) is measured across different configurations. The measurements are conducted on 500 randomly selected images from the COCO test set using the prompt “Describe the image in detail.” with a maximum of 256 newly generated tokens per image. The time cost of `model.forward` is recorded for both prefill and decode phases, and the results for prefill time, decode throughput (tokens per second), and FLOPs are presented in Table 7.

The results reveal that computational cost scales approximately linearly with vision token count. Specifically, FLOPs[6] increase from 2.380 TFLOPs to 4.151 TFLOPs as n_l increases from 1 to 768, and the same trend is observed in prefill time. In contrast, decode throughput remains relatively stable across different n_l values due to the KV cache mechanism. During

---

[6] FLOPs are measured using the library `thop`: `https://github.com/Lyken17/pytorch-OpCounter`

Table 5: Comparison of token reduction methods. VisPruner (Zhang et al., 2024) is configured with dominant $n_l - 10$, contextual token 10, and base model LLaVA-v1.5-7B. VisionZip (Yang et al., 2025) is configured with visual token num n_l, important ratio 0.5, and base model LLaVA-v1.5-7B. The proposed method is configured with n_l, and use "Vision Question Queries" as input. The largest value for each setting is marked **bold**.

Methods	MME	OCRBench	HallusionBench	POPE	AI2D TEST	RealWorldQA	MMStar	ScienceQA TEST	OCRVQA TESTCORE	ChartQA TEST	TextVQA VAL	SEEDBench IMG	SEEDBench2 Plus	BLEU-1 COCO VAL	BLEU-4 COCO VAL	ROUGE-L COCO VAL	CIDEr COCO VAL
$n_l = 512$																	
VisPruner	**1691.82**	308.00	23.11	81.19	51.88	48.10	32.00	**68.02**	**52.90**	14.84	22.76	**60.15**	38.87	21.19	5.13	21.30	0.95
VisionZip	1527.86	303.00	19.76	69.75	**51.98**	48.24	31.67	67.58	41.08	12.88	20.26	59.63	38.95	21.36	5.19	21.56	1.18
Proposed	1414.27	**472.00**	**29.56**	**86.63**	40.84	**49.15**	**35.40**	57.16	38.02	**51.48**	**53.55**	53.34	**43.39**	**40.71**	**10.57**	**34.70**	**15.26**
$n_l = 384$																	
VisPruner	**1682.13**	312.00	24.39	84.59	51.30	**49.28**	31.93	**68.17**	**54.65**	14.88	25.29	**59.89**	38.91	21.32	5.21	21.65	1.49
VisionZip	1590.55	295.00	20.43	77.56	**51.65**	48.63	32.73	67.77	47.49	13.16	20.77	59.46	38.47	21.30	5.21	21.55	1.19
Proposed	1409.44	**445.00**	**32.81**	**85.25**	39.41	40.00	**35.47**	53.84	30.79	**43.20**	**53.40**	52.07	**44.05**	**38.04**	**9.74**	**32.42**	**7.76**
$n_l = 256$																	
VisPruner	**1551.58**	302.00	26.33	84.88	51.04	47.71	32.00	**67.72**	**52.51**	13.84	23.61	**59.14**	38.30	22.05	5.44	22.70	2.85
VisionZip	1535.73	299.00	20.74	80.91	**52.27**	**49.02**	31.73	67.43	47.14	12.88	19.84	58.62	38.91	22.28	5.47	22.87	2.85
Proposed	1472.53	**448.00**	**27.51**	**86.73**	39.48	45.62	**34.13**	52.90	43.29	**50.60**	**54.40**	52.11	**39.44**	**38.79**	**9.84**	**33.04**	**9.11**
$n_l = 128$																	
VisPruner	**1605.53**	291.00	24.19	**84.61**	51.00	**47.71**	**30.40**	**68.32**	**42.02**	12.48	19.77	**57.66**	38.30	25.87	6.57	28.26	15.76
VisionZip	1529.75	286.00	21.97	82.53	**51.59**	46.93	29.33	68.02	34.11	11.64	17.20	56.89	**38.60**	26.92	6.78	29.26	**17.30**
Proposed	1396.15	**438.00**	**29.30**	84.02	33.61	45.88	30.33	48.44	38.61	**41.32**	**49.47**	47.63	27.80	**37.19**	**9.38**	**31.46**	5.49
$n_l = 64$																	
VisPruner	**1536.95**	283.00	21.54	80.32	49.68	45.62	29.27	67.28	25.52	10.72	16.30	**55.54**	37.99	30.25	7.76	32.85	27.24
VisionZip	1509.86	273.00	20.45	77.80	**50.42**	47.06	28.67	**67.63**	21.81	11.04	15.87	55.27	**38.95**	33.46	8.78	**34.80**	**32.47**
Proposed	1328.09	**386.00**	**26.65**	**82.76**	35.04	**49.67**	**31.73**	49.28	**41.70**	**38.16**	**45.80**	47.83	28.37	**37.05**	**9.21**	31.35	6.70
$n_l = 32$																	
VisPruner	1379.74	239.00	19.92	73.10	48.15	44.84	26.87	**67.43**	14.94	10.16	13.03	51.24	38.56	38.56	9.77	36.75	37.08
VisionZip	**1419.03**	224.00	20.12	66.95	**49.09**	43.14	28.00	66.63	21.42	11.08	13.07	**51.66**	**38.65**	**45.69**	**12.52**	**39.02**	**44.58**
Proposed	1321.83	**320.00**	**20.81**	**81.23**	35.46	**49.41**	**31.80**	50.27	**30.92**	**29.68**	**42.60**	45.03	30.35	35.47	8.11	29.85	5.74
$n_l = 16$																	
VisPruner	1203.98	150.00	18.65	49.75	46.89	41.70	26.20	**66.14**	16.44	9.40	9.36	**46.72**	**38.16**	**45.12**	10.69	**35.76**	**34.66**
VisionZip	1076.22	64.00	17.77	36.26	**47.22**	38.43	24.93	64.70	**25.10**	10.68	7.21	44.70	37.07	44.02	**11.36**	30.48	28.54
Proposed	**1291.08**	**268.00**	**21.27**	**79.53**	36.30	**47.71**	**30.80**	47.25	21.81	**20.32**	**34.11**	42.85	31.80	37.61	8.75	32.33	15.51

the decode phase, attention computations are substantially reduced because only the newly generated token attends to the cached key-value pairs from previous tokens, including vision tokens. Consequently, variations in n_l from 1 to 768 do not significantly impact attention computation costs during autoregressive generation, resulting in consistent tokens-per-second rates across configurations.

Balancing the trade-off between computational efficiency and performance is crucial for practical deployment, particularly in resource-constrained environments or latency-sensitive

Table 6: Comparison of time to first token (TTFT), GPU utilization, and GPU memory usage. The proposed method is configured with n_l, and use "Vision Question Queries" as input. The GPU utilization and memory usage are measured on a single NVIDIA RTX 3090 GPU (24G). Note: TTFTs below are measured from image loading, tokenization to first token generation.

n_l	TTFT (s)	GPU util (%)	GPU peak (GiB)	GPU average (GiB)
768	0.3038	81.9693	15.4844	15.4832
512	0.3024	83.8838	15.4844	15.4830
256	0.3013	83.9863	15.4844	15.4830
128	0.2554	79.9462	15.4258	15.4244
64	0.2228	73.8923	15.4786	15.4770
32	0.2203	74.1203	15.4454	15.4439
16	0.2211	76.7657	15.5391	15.5373
8	0.2181	75.9544	15.4005	15.3990
1	0.2137	73.0568	15.4395	15.4338

applications. Measuring the time cost can provide valuable insights into the computational trade-offs associated with different vision token counts, helping to inform the selection of an optimal configuration for specific use cases.

Table 7: Comparison of prefill time, tokens per second in decode phase and TFLOPs of the prefill. The proposed method is configured with n_l, and use "Vision Question Queries" as input. The GPU utilization and memory usage are measured on a single NVIDIA RTX 3090 GPU (24G). Note: the data below are measured on `model.forward`; FLOPs will be different due to different size of input images (A high resolution image will be cropped into nine patches. If its size is not big enough ($336 \leq$ width or height $\leq 336 \times 3$), the patches without any pixels will be ignored.); all data cells are format as (mean ± std).

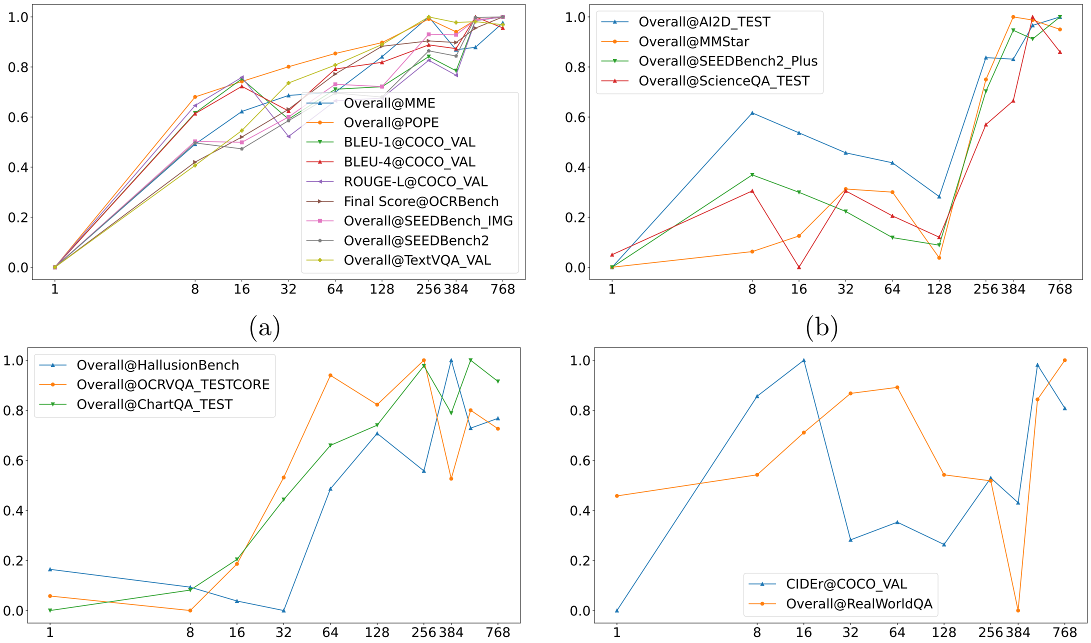
n_l	Prefill Time (s)	Tokens/s	TFLOPs ($1e^{12}$)
768	0.255 ± 0.023	23.114 ± 0.103	4.151 ± 0.326
512	0.255 ± 0.023	23.092 ± 0.113	4.151 ± 0.326
384	0.285 ± 0.020	23.032 ± 0.088	5.040 ± 0.326
256	0.255 ± 0.023	23.100 ± 0.099	4.151 ± 0.326
128	0.211 ± 0.022	23.049 ± 0.083	3.262 ± 0.326
64	0.178 ± 0.022	23.051 ± 0.130	2.817 ± 0.326
32	0.177 ± 0.022	23.067 ± 0.115	2.595 ± 0.326
16	0.177 ± 0.022	23.141 ± 0.124	2.484 ± 0.326
8	0.175 ± 0.022	23.188 ± 0.121	2.428 ± 0.326
1	0.172 ± 0.029	23.080 ± 0.145	2.380 ± 0.326

Appendix F. Further Analysis on n_l

The following analysis focuses on the general trends and patterns observed across different experimental configurations.

To facilitate comprehensive comparison across benchmarks with varying scales, the results are normalized using min-max scaling: $\mathbf{x} \leftarrow (\mathbf{x} - \min(\mathbf{x}))/(\max(\mathbf{x}) - \min(\mathbf{x}))$, which transforms all results to the range $[0, 1]$. This normalization enables consistent comparison of performance trends across diverse benchmarks while preserving the relative relationships within each dataset. The normalized results reveal several distinct behavioral patterns when analyzed as functions of n_l. To characterize these patterns and better illustration, K-means clustering is applied to partition the results into four clusters, providing a better visualization for observing how n_l influences model performance and identifying optimal configurations across different scenarios.

Analysis with vision tokens, questions, and learnable queries. The analysis begins with the model configuration that processes vision tokens, user questions, and learnable queries as inputs. The results, illustrated in Figure 18, reveal distinct clusters that characterize the relationship between the number of learnable queries (n_l) and model performance.



(a) (b) (c) (d)

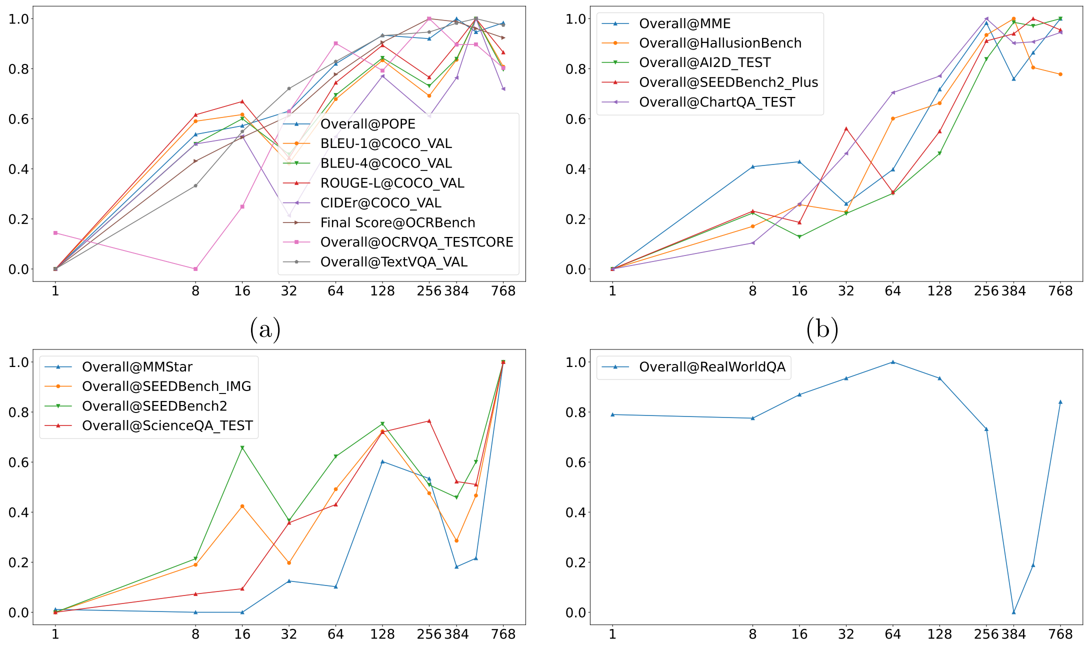
Figure 18: Clustered view of performances. Inputs: vision tokens, user questions, and learnable queries. Values are normalized using min-max scaling. X-axis is $\log_2(n_l)$.

A consistent observation across all clusters is that model performance remains suboptimal when n_l is small. As n_l increases, performance improves substantially before eventually stabilizing at higher values. This pattern demonstrates that n_l serves as a critical factor in enabling the model to achieve optimal performance.

Figure 18a demonstrates an approximately linear relationship between performance and $\log_2(n_l)$, indicating consistent logarithmic scaling behavior within this cluster. In Figure 18b, the model exhibits degraded performance for $n_l < 128$, followed by rapid improvement as n_l increases, ultimately achieving stability at higher values. Similarly, Figure 18c shows poor performance for $n_l < 64$, with subsequent rapid improvement and stabilization as n_l grows. These observations reinforce the critical importance of sufficient n_l for achieving optimal model performance. However, the behavior observed in Figure 18d exhibits relatively noisy behavior. There is a sharp drop in performance at specific n_l values, followed by a gradual recovery. This pattern suggests that the under specific tasks or datasets, there could be unexpected noise or instability in the model performance.

Analysis without user questions. To examine the impact of incorporating user text questions on model performance, an alternative configuration is evaluated where the model excludes user questions as input. In this setup, the model processes only vision tokens and learnable queries. The model described in Appendix C is fine-tuned under this modified configuration. The resulting performance trends are presented in Figure 19, which reveals four distinct clusters with varying patterns relative to n_l.

Figures 19a and 19b demonstrate behavior approximately proportional to $\log_2(n_l)$, indicating consistent logarithmic scaling trends within these clusters. Figure 19c exhibit less consistent linear relationships, however the performance still improves with increasing n_l. Figure 19d is an exception, showing a sharp drop in performance at $n_l = 128$. However, if this point is excluded, the rest of the data can show a logarithmic trend.



(a) (b)

(c) (d)

Figure 19: Clustered view of performances. Inputs: vision tokens, learnable queries (excluding user questions). Values are normalized using min-max scaling. X-axis is $\log_2(n_l)$.

Appendix G. Example of Inference

The following illustrate two demonstrations of the generation results produced by the models.

G.1 Example 1

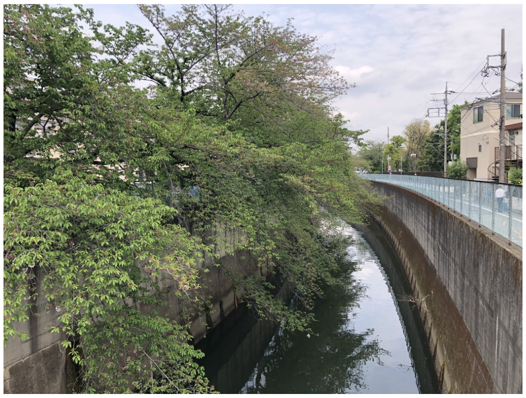


Figure 20: Image of example 1.

User's question: "Please describe the image in full detail."
Configuration $n_l = 1$:

The image is of a small river flowing through a green landscape. The river is surrounded by trees and rocks, and there are several bridges crossing it. The river is also lined with rocks and greenery.

Configuration $n_l = 8$:

The image is of a river flowing through a city. There are trees and buildings along the riverbank. The river is surrounded by a concrete wall. There are also some rocks in the river.

Configuration $n_l = 16$:

The image is of a river flowing through a concrete channel. The river is surrounded by trees and buildings. There are also some houses and a bridge in the background.

Configuration $n_l = 32$:

The image shows a long, narrow, concrete drainage ditch running along a wall of a building. The ditch is surrounded by trees and greenery, and there are people walking along the wall. The wall is made of brick and has a large pipe running along it.

Configuration $n_l = 64$:

The image is of a canal or river with a wall on one side and a building on the other. There are trees and greenery along the wall.

Configuration $n_l = 128$:

A canal is surrounded by a wall and has a green tree growing on it.

Configuration $n_l = 256$:

The image is a scenic view of a canal, with a wall on the left and a fence on the right. The canal is surrounded by trees, with leaves and branches visible. There are also houses and buildings visible in the background. The image is quite detailed, with a lot of elements visible.

Configuration $n_l = 384$:

A canal with a concrete wall and a tree growing on the side.

Configuration $n_l = 512$:

The image shows a scene of a canal with a wall on the side, surrounded by trees and houses. The canal is filled with water, and there are reflections of the trees and houses on the water. The wall is made of concrete, and there are houses on the other side of the canal.

Configuration $n_l = 768$:

The image shows a scene of a canal, with a wall of trees on one side and a fence on the other. The canal is surrounded by a concrete wall, and there are trees growing on the wall. The trees are lush and green, with leaves and branches visible. There are also people walking along the fence, adding to the scene's vibrancy.

Baseline model LLaVA-v1.5-7B-hf:

The image features a small stream of water flowing through a city, surrounded by trees and buildings. The stream is lined with trees, creating a serene atmosphere. There are several people scattered throughout the scene, some closer to the water and others further away.
In the background, a car can be seen parked near the stream, indicating that the area is accessible to pedestrians and vehicles. The combination of the waterway, trees, and people creates a picturesque urban scene.

Baseline model InstructBLIP-7B-hf:

The image features a narrow canal or river running alongside a building, with trees lining the water's edge. There are several trees visible along the canal, with some closer to the water and others further away. The trees are spread out along the canal, creating a serene and peaceful atmosphere. A few cars can be seen parked along the side of the canal, adding to the urban setting.

G.2 Example 2

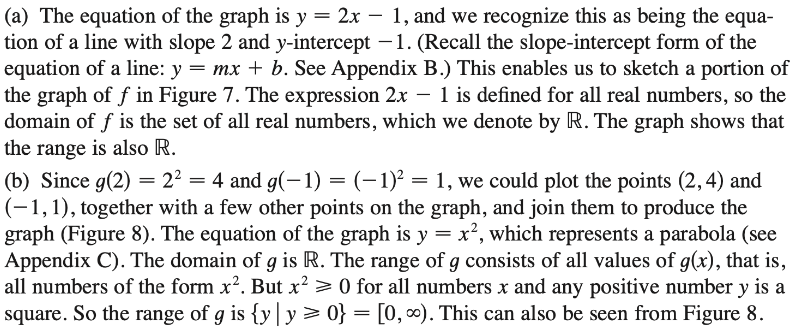
User's question: "Please read all text from the image."

Configuration $n_l = 1$:

- -

Configuration $n_l = 8$:

x 2 = 0.
The equation is a linear equation in two variables, x and y.



(a) The equation of the graph is $y = 2x - 1$, and we recognize this as being the equation of a line with slope 2 and y-intercept -1. (Recall the slope-intercept form of the equation of a line: $y = mx + b$. See Appendix B.) This enables us to sketch a portion of the graph of f in Figure 7. The expression $2x - 1$ is defined for all real numbers, so the domain of f is the set of all real numbers, which we denote by $\mathbb{R}$. The graph shows that the range is also $\mathbb{R}$.

(b) Since $g(2) = 2^2 = 4$ and $g(-1) = (-1)^2 = 1$, we could plot the points $(2, 4)$ and $(-1, 1)$, together with a few other points on the graph, and join them to produce the graph (Figure 8). The equation of the graph is $y = x^2$, which represents a parabola (see Appendix C). The domain of g is $\mathbb{R}$. The range of g consists of all values of $g(x)$, that is, all numbers of the form x^2. But $x^2 \geqslant 0$ for all numbers x and any positive number y is a square. So the range of g is $\{y \mid y \geqslant 0\} = [0, \infty)$. This can also be seen from Figure 8.

Figure 21: Image of example 2.

```
The graph of the equation is a straight line.
The equation is a function of x and y.
The equation is a function of
```

Configuration $n_l = 16$:

```
The equation of the line is:
\(y=m\*x+b\),
where m is the slope of the line and b is the y−intercept.
The slope of the line is the ratio of the change in the y−coordinate to the change in the x−coordinate. In other words, it is the rate at which the line rises or falls. The y−intercept is the point at which the line crosses the y−axis.
The slope of a line can be calculated by dividing the change in the y−coordinate by the change in the x−coordinate. For example, if the line passes through the point (2, 3), the slope is 3/2.
The y−intercept can be calculated by adding the slope to the point at which the line crosses the y−axis. For example, if the line passes through the point (2, 3), the y−intercept is 3.
The equation of a line can be written in the form:
\(y=m\*x+b\),
where m is the slope and b is the y−intercept.
The equation of a line can also be written in the form:
\(y=m\*x+c\),
where m
```

Configuration $n_l = 32$:

```
## 3.1.1 The equation is:
\begin{aligned}
\text{the equation is} &= \text{as we are expecting} \\
\text{and} &= \text{the slope of the line is} \\
\text{the slope of the line is} &= \text{the slope of the line is} \\
\text{the slope of the line is} &= \text{the slope of the line is
```

Configuration $n_l = 64$:

## The equation of a line with slope and y−intercept
The equation of a line: y = mx + b. + See Appendix B. + See Appendix C. + See Appendix D.
The graph of the equation in Figure 7.2 is defined for all values of x. The expression x = 0 is defined for all values of y. The range of the graph is also R.
Since g = 2 since g = 1 and g = −1, we can write the equation of the line as:
y = mx + b. + See Appendix B. + See Appendix C. + See Appendix D.
The equation of a line with slope and y−intercept is a linear equation. The slope of the line is m, and the y−intercept is b. The equation of a line with slope and y−intercept is a linear equation. The slope of the line is m, and the y−intercept is b.
The equation of a line with slope and y−intercept is a linear

Configuration $n_l = 128$:

The equation of a line with slope s and a point (x,y) is 2x − y = s. We recognize this as the equation of a line in slope−intercept form of a line: mx + b = x. See Appendix B. This enables us to sketch a portion of the graph of f in Figure 7. The expression 2 is defined for all real numbers, so the domain of f is the set of all real numbers, which we denote by R. The graph shows that so is also R. (2) Since g(2) = s since g(2) = s, we could plot the points (2,4) and (1,1) on the graph. (1,−1), 1, 1,

Configuration $n_l = 256$:

The equation of the graph is y = 2x − 1, and we recognize this as being the equation of a line with slope 2 and y−intercept −1. (Recall the slope−intercept form of the equation of a line: y = mx + b. See Appendix B.) This enables us to sketch a portion of the graph of f in Figure 7. The expression 2x − 1 is defined for all real numbers, so the domain of f is the set of all real numbers, which we denote by R. The graph shows that the range is also R. \(\begin{array}{c} \text{Since} \ g(2)=2=1\text{, we could plot the points (2,4) and} \\ \text{together, with a few other points on the graph, and join them to produce the graph (Figure 8). The equation of the graph is y = x2, which represents a parabola (see Appendix C). The domain of g is R. The range of g consists of all values of g(x), that is, all numbers of the form x2. But x2 > 0 for any positive number y is a square. So the range of g is [y | y | y] = 0. This can also be seen from Figure 8.

Configuration $n_l = 384$:

(a) The equation of the graph is y = 2x − 1, and we recognize this as being the equation of a line with slope 2 and y−intercept (1. Recall the slope−intercept form of the equation of a line: y = mx + b. See Appendix B.) This enables us to sketch a portion of the graph of f in Figure 7. The expression 2x − 1 is defined for all real numbers, so the domain of f is the set of all real numbers, which we denote by R. The graph shows that the range is also R. (b) Since g(2) = 2a = −1, we could plot the points (2, 4) and (1, 1), together with a few other points on the graph, and join them to produce the graph (Figure 8). The equation of the graph is y = x2, which represents a parabola (see Appendix C). The domain of g is R. The range of g consists of all values of g(x), that is, all numbers of the form x2. But x2 > 0 for all numbers and any positive number y is a. So the range of g is [0, y] = [0, y] = [0, y] = [0, y] =

Configuration $n_l = 512$:

∗ The equation of the graph is y = 2x − 1, and we recognize this as being the equation of a line with slope 2 and y−intercept −1. (Recall the slope−intercept form of the equation of a line: y = mx + b. See Appendix B.) This enables us to sketch a portion of the graph of f in Figure 7. The expression 2x − 1 is defined for all real numbers, so the domain of f is the set of all real numbers, which we denote by R. The graph shows that the range is also R. (b) Since g(2) = 2 = 1, we could plot the points (2, 4) and (1, −1), together with a few other points on the graph, and join them to produce the graph (Figure 8). The equation of the graph is y = x2, which represents a parabola (see Appendix C). The domain of g is R. The range of g consists of all numbers of the form x2. But x2 > 0 for any positive number y is a square. So the range of g is y = 0 = 0, or y = 0 = 0. This can also be seen from Figure 8.

Configuration $n_l = 768$:

<table> <thead>
<tr>
<td>The equation of the graph is y = 2x − 1, and we recognize this as being the equation of a line with slope 2 and y−intercept −1. (Recall the slope−intercept form of the equation of a line: y = mx + b. See Appendix B.) This enables us to sketch a portion of the graph of f in Figure 7. The expression 2x − 1 is defined for all real numbers, so the domain of f is the set of all real numbers, which we denote by R. The graph shows that the range is also R. (b) Since g(2) = 2 = (1 − 1) = (1 − 1), together with a few other points on the graph, and join them to produce the graph (Figure 8). The equation of the graph is y = x2, which represents a parabola (see Appendix C). The domain of g is R. The range of g consists of all values of g, that is, all numbers of the form x2. But x2 > 0 for any positive number y is a square. So the range of g is [y | y | y] = [0 | 0 | 0]. This can also be seen from Figure 8. </td> <td>The equation of the graph is y = 2x − 1, and we recognize this as being the equation of a line with slope 2 and y−intercept −1. (Recall the slope−intercept form of the equation of a line: y = mx + b. See Appendix B.) This enables us to sketch a portion of the graph of f in Figure 7. The expression 2x − 1 is defined for all real numbers, so the domain of f is the set of all real numbers, which we denote by R. The graph shows that the range is also R. (b) Since g(2) = 2 = (1 − 1) = (1 − 1), together with a few other points on the graph, and join them to produce the graph (Figure 8). The equation of the graph is y = x2, which represents a parabola (see Appendix C). The domain of g is R. The range of g consists of all values of g, that

Baseline model LLaVA-v1.5-7B-hf:

The image displays a graph with a slope, and the equation is written below it. The graph is described as a ”real number” graph, and the slope is defined as a ”real number.” The equation is written in terms of the slope, and the graph is described as a ”real number” graph. The slope is also described as a ”real number.” The equation is written in terms of the slope, and the graph is described as a ”real number” graph. The slope is also described as a

Baseline model InstructBLIP-7B-hf:

the example of graph is shown in the following

References

Marah I Abdin, Sam Ade Jacobs, Ammar Ahmad Awan, Jyoti Aneja, Ahmed Awadallah, Hany Awadalla, et al. Phi-3 technical report: A highly capable language model locally on your phone. CoRR, abs/2404.14219, 2024. doi: 10.48550/ARXIV.2404.14219. URL `https://doi.org/10.48550/arXiv.2404.14219`.

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, et al. Flamingo: a visual language model for few-shot learning. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh, editors, Annual Conference on Neural Information Processing Systems, NeurIPS, New Orleans, LA, USA, November 28–December 9, 2022, 2022. URL `http://papers.nips.cc/paper_files/paper/2022/hash/960a172bc7fbf0177ccccbb411a7d800-Abstract-Conference.html`.

Anthropic. The claude 3 model family: Opus, sonnet, haiku. `https://www-cdn.anthropic.com/de8ba9b01c9ab7cbabf5c33b80b7bbc618857627/Model_Card_Claude_3.pdf`, 2024. Large language model.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, et al. Qwen technical report. CoRR, abs/2309.16609, 2023a. doi: 10.48550/ARXIV.2309.16609. URL `https://doi.org/10.48550/arXiv.2309.16609`.

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. CoRR, abs/2308.12966, 2023b. doi: 10.48550/ARXIV.2308.12966. URL `https://doi.org/10.48550/arXiv.2308.12966`.

Rohan Bavishi, Erich Elsen, Curtis Hawthorne, Maxwell Nye, Augustus Odena, Arushi Somani, and Sağnak Taşırlar. Introducing our multimodal models, 2023. URL `https://www.adept.ai/blog/fuyu-8b`.

Lukas Blecher, Guillem Cucurull, Thomas Scialom, and Robert Stojnic. Nougat: Neural optical understanding for academic documents. In International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024. OpenReview.net, 2024. URL `https://openreview.net/forum?id=fUtxNAKpdV`.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, et al. Language models are few-shot learners. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, Annual Conference on Neural Information Processing Systems, NeurIPS, December 6-12, 2020, virtual, 2020. URL `https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html`.

Lin Chen, Jinsong Li, Xiaoyi Dong, Pan Zhang, Yuhang Zang, Zehui Chen, et al. Are we on the right way for evaluating large vision-language models? CoRR, abs/2403.20330, 2024a. doi: 10.48550/ARXIV.2403.20330. URL `https://doi.org/10.48550/arXiv.2403.20330`.

Zhe Chen, Jiannan Wu, Wenhai Wang, Weijie Su, Guo Chen, Sen Xing, 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, 2024b.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, et al. Vicuna: An open-source chatbot impressing gpt-4 with 90% chatgpt quality, March 2023. URL `https://lmsys.org/blog/2023-03-30-vicuna/`.

Wenliang Dai, Junnan Li, Dongxu Li, Anthony Meng Huat Tiong, Junqi Zhao, Weisheng Wang, et al. Instructblip: Towards general-purpose vision-language models with instruction tuning. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, Annual Conference on Neural Information Processing Systems, NeurIPS, New Orleans, LA, USA, December 10 - 16, 2023, 2023. URL `http://papers.nips.cc/paper_files/paper/2023/hash/9a6a435e75419a836fe47ab6793623e6-Abstract-Conference.html`.

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. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, Advances in Neural Information Processing Systems, volume 37, pages 42566–42592. Curran Associates, Inc., 2024. doi: 10.52202/079017-1348. URL `https://proceedings.neurips.cc/paper_files/paper/2024/file/4b06cdddb1cde6624c0be1465c7b800f-Paper-Conference.pdf`.

Haodong Duan, Junming Yang, Yuxuan Qiao, Xinyu Fang, Lin Chen, Yuan Liu, et al. Vlmevalkit: An open-source toolkit for evaluating large multi-modality models. In Jianfei Cai, Mohan S. Kankanhalli, Balakrishnan Prabhakaran, Susanne Boll, Ramanathan Subramanian, Liang Zheng, Vivek K. Singh, Pablo César, Lexing Xie, and Dong Xu, editors, ACM International Conference on Multimedia, MM 2024, Melbourne, VIC, Australia, 28 October 2024 - 1 November 2024, pages 11198–11201. ACM, 2024. doi: 10.1145/3664647.3685520. URL `https://doi.org/10.1145/3664647.3685520`.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, et al. The llama 3 herd of models. CoRR, abs/2407.21783, 2024. doi: 10.48550/ARXIV.2407.21783. URL `https://doi.org/10.48550/arXiv.2407.21783`.

Chaoyou Fu, Peixian Chen, Yunhang Shen, Yulei Qin, Mengdan Zhang, Xu Lin, Jinrui Yang, Xiawu Zheng, Ke Li, Xing Sun, et al. MME: A comprehensive evaluation benchmark for multimodal large language models. In The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track, 2025.

Tianrui Guan, Fuxiao Liu, Xiyang Wu, Ruiqi Xian, Zongxia Li, Xiaoyu Liu, et al. Hallusionbench: An advanced diagnostic suite for entangled language hallucination and visual illusion in large vision-language models. In IEEE/CVF Conference on

Computer Vision and Pattern Recognition, CVPR, Seattle, WA, USA, June 16-22, 2024, pages 14375–14385. IEEE, 2024. doi: 10.1109/CVPR52733.2024.01363. URL https://doi.org/10.1109/CVPR52733.2024.01363.

Zonghao Guo, Ruyi Xu, Yuan Yao, Junbo Cui, Zanlin Ni, Chunjiang Ge, et al. Llava-uhd: An LMM perceiving any aspect ratio and high-resolution images. In Ales Leonardis, Elisa Ricci, Stefan Roth, Olga Russakovsky, Torsten Sattler, and Gül Varol, editors, Computer Vision ECCV, Milan, Italy, September 29-October 4, 2024, volume 15141 of Lecture Notes in Computer Science, pages 390–406. Springer, 2024. doi: 10.1007/978-3-031-73010-8_23. URL https://doi.org/10.1007/978-3-031-73010-8_23.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, et al. Training compute-optimal large language models, 2022. URL https://arxiv.org/abs/2203.15556.

Anwen Hu, Haiyang Xu, Jiabo Ye, Ming Yan, Liang Zhang, Bo Zhang, et al. mplug-docowl 1.5: Unified structure learning for ocr-free document understanding. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, Findings of the Association for Computational Linguistics: EMNLP, Miami, Florida, USA, November 12-16, 2024, pages 3096–3120. Association for Computational Linguistics, 2024. URL https://aclanthology.org/2024.findings-emnlp.175.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, et al. Scaling laws for neural language models. CoRR, abs/2001.08361, 2020. URL https://arxiv.org/abs/2001.08361.

Bohao Li, Yuying Ge, Yixiao Ge, Guangzhi Wang, Rui Wang, Ruimao Zhang, and Ying Shan. Seed-bench-2: Benchmarking multimodal large language models. CoRR, abs/2311.17092, 2023a. doi: 10.48550/ARXIV.2311.17092. URL https://doi.org/10.48550/arXiv.2311.17092.

Bohao Li, Yuying Ge, Yi Chen, Yixiao Ge, Ruimao Zhang, and Ying Shan. Seed-bench-2-plus: Benchmarking multimodal large language models with text-rich visual comprehension. CoRR, abs/2404.16790, 2024a. doi: 10.48550/ARXIV.2404.16790. URL https://doi.org/10.48550/arXiv.2404.16790.

Bohao Li, Yuying Ge, Yixiao Ge, Guangzhi Wang, Rui Wang, Ruimao Zhang, and Ying Shan. Seed-bench: Benchmarking multimodal large language models. In IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2024, Seattle, WA, USA, June 16-22, 2024, pages 13299–13308. IEEE, 2024b. doi: 10.1109/CVPR52733.2024.01263. URL https://doi.org/10.1109/CVPR52733.2024.01263.

Junnan Li, Dongxu Li, Silvio Savarese, and Steven C. H. Hoi. BLIP-2: bootstrapping language-image pre-training with frozen image encoders and large language models. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, International Conference on Machine Learning, ICML, 23-29 July 2023, Honolulu, Hawaii, USA, volume 202 of Proceedings of Machine Learning Research, pages 19730–19742. PMLR, 2023b. URL https://proceedings.mlr.press/v202/li23q.html.

Kevin Li, Sachin Goyal, João D Semedo, and J Zico Kolter. Inference optimal vlms need fewer visual tokens and more parameters. In The Thirteenth International Conference on Learning Representations, 2024c.

Yifan Li, Yifan Du, Kun Zhou, Jinpeng Wang, Wayne Xin Zhao, and Ji-Rong Wen. Evaluating object hallucination in large vision-language models. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023, pages 292–305. Association for Computational Linguistics, 2023c. doi: 10.18653/V1/2023.EMNLP-MAIN.20. URL `https://doi.org/10.18653/v1/2023.emnlp-main.20`.

Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, et al. Microsoft COCO: common objects in context. In David J. Fleet, Tomás Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, volume 8693 of Lecture Notes in Computer Science, pages 740–755. Springer, 2014. doi: 10.1007/978-3-319-10602-1_48. URL `https://doi.org/10.1007/978-3-319-10602-1_48`.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, Annual Conference on Neural Information Processing Systems, NeurIPS, New Orleans, LA, USA, December 10–16, 2023, 2023. URL `http://papers.nips.cc/paper_files/paper/2023/hash/6dcf277ea32ce3288914faf369fe6de0-Abstract-Conference.html`.

Yuliang Liu, Zhang Li, Mingxin Huang, Biao Yang, Wenwen Yu, Chunyuan Li, Xu-Cheng Yin, Cheng-Lin Liu, Lianwen Jin, and Xiang Bai. Ocrbench: on the hidden mystery of ocr in large multimodal models. Science China Information Sciences, 67(12):220102, 2024.

Haoyu Lu, Wen Liu, Bo Zhang, Bingxuan Wang, Kai Dong, Bo Liu, et al. Deepseek-vl: Towards real-world vision-language understanding. CoRR, abs/2403.05525, 2024. doi: 10.48550/ARXIV.2403.05525. URL `https://doi.org/10.48550/arXiv.2403.05525`.

Pan Lu, Swaroop Mishra, Tanglin Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, et al. Learn to explain: Multimodal reasoning via thought chains for science question answering. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh, editors, Annual Conference on Neural Information Processing Systems, NeurIPS, New Orleans, LA, USA, November 28 - December 9, 2022, 2022. URL `http://papers.nips.cc/paper_files/paper/2022/hash/11332b6b6cf4485b84afadb1352d3a9a-Abstract-Conference.html`.

Ahmed Masry, Do Long, Jia Qing Tan, Shafiq Joty, and Enamul Hoque. ChartQA: A benchmark for question answering about charts with visual and logical reasoning. In Findings of the Association for Computational Linguistics: ACL 2022, pages 2263–2279, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-acl.177. URL `https://aclanthology.org/2022.findings-acl.177`.

Anand Mishra, Shashank Shekhar, Ajeet Kumar Singh, and Anirban Chakraborty. OCR-VQA: visual question answering by reading text in images. In International Conference on Document Analysis and Recognition, ICDAR 2019, Sydney, Australia, September 20-25, 2019, pages 947–952. IEEE, 2019. doi: 10.1109/ICDAR.2019.00156. URL `https://doi.org/10.1109/ICDAR.2019.00156`.

Niklas Muennighoff, Alexander M. Rush, Boaz Barak, Teven Le Scao, Aleksandra Piktus, Nouamane Tazi, Sampo Pyysalo, Thomas Wolf, and Colin Raffel. Scaling data-constrained language models. Journal of Machine Learning Research, 26(53):1–66, 2025. URL `http://jmlr.org/papers/v26/24-1000.html`.

OpenAI. GPT-4 technical report. CoRR, abs/2303.08774, 2023. doi: 10.48550/ARXIV.2303.08774. URL `https://doi.org/10.48550/arXiv.2303.08774`.

OpenAI. Gpt-4o system card, 2024. URL `https://arxiv.org/abs/2410.21276`.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, and Sandothers. Learning transferable visual models from natural language supervision. In Marina Meila and Tong Zhang, editors, International Conference on Machine Learning, ICML, 18-24 July 2021, Virtual Event, volume 139 of Proceedings of Machine Learning Research, pages 8748–8763. PMLR, 2021. URL `http://proceedings.mlr.press/v139/radford21a.html`.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D. Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, Annual Conference on Neural Information Processing Systems, NeurIPS, New Orleans, LA, USA, December 10–16, 2023, 2023. URL `http://papers.nips.cc/paper_files/paper/2023/hash/a85b405ed65c6477a4fe8302b5e06ce7-Abstract-Conference.html`.

Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan, editors, Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019, pages 3980–3990. Association for Computational Linguistics, 2019. doi: 10.18653/V1/D19-1410. URL `https://doi.org/10.18653/v1/D19-1410`.

Amanpreet Singh, Vivek Natarajan, Meet Shah, Yu Jiang, Xinlei Chen, Dhruv Batra, et al. Towards VQA models that can read. In IEEE Conference on Computer Vision and Pattern Recognition, CVPR, Long Beach, CA, USA, June 16-20, 2019, pages 8317–8326. Computer Vision Foundation / IEEE, 2019. doi: 10.1109/CVPR.2019.00851. URL `http://openaccess.thecvf.com/content_CVPR_2019/html/Singh_Towards_VQA_Models_That_Can_Read_CVPR_2019_paper.html`.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, et al. Llama 2: Open foundation and fine-tuned chat models. CoRR, abs/2307.09288, 2023. doi: 10.48550/ARXIV.2307.09288. URL `https://doi.org/10.48550/arXiv.2307.09288`.

Weihan Wang, Qingsong Lv, Wenmeng Yu, Wenyi Hong, Ji Qi, Yan Wang, Junhui Ji, Zhuoyi Yang, Lei Zhao, Song XiXuan, et al. Cogvlm: Visual expert for pretrained language models. Advances in Neural Information Processing Systems, 37:121475–121499, 2024.

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

Jiabo Ye, Anwen Hu, Haiyang Xu, Qinghao Ye, Ming Yan, Yuhao Dan, et al. mplug-docowl: Modularized multimodal large language model for document understanding. CoRR, abs/2307.02499, 2023. doi: 10.48550/ARXIV.2307.02499. URL `https://doi.org/10.48550/arXiv.2307.02499`.

Qinghao Ye, Haiyang Xu, Jiabo Ye, Ming Yan, Anwen Hu, Haowei Liu, Qi Qian, Ji Zhang, and Fei Huang. mPLUG-Owl2: Revolutionizing multi-modal large language model with modality collaboration. In Proceedings of the ieee/cvf conference on computer vision and pattern recognition, pages 13040–13051, 2024.

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

Qizhe Zhang, Aosong Cheng, Ming Lu, Zhiyong Zhuo, Minqi Wang, Jiajun Cao, Shaobo Guo, Qi She, and Shanghang Zhang. [CLS] attention is all you need for training-free visual token pruning: Make vlm inference faster. arXiv preprint arXiv:2412.01818, 2024.

Bo Zhao, Boya Wu, and Tiejun Huang. SVIT: scaling up visual instruction tuning. CoRR, abs/2307.04087, 2023. doi: 10.48550/ARXIV.2307.04087. URL `https://doi.org/10.48550/arXiv.2307.04087`.

Deyao Zhu, Jun Chen, Xiaoqian Shen, Xiang Li, and Mohamed Elhoseiny. Minigpt-4: Enhancing vision-language understanding with advanced large language models. In International Conference on Learning Representations, ICLR, Vienna, Austria, May 7-11, 2024. OpenReview.net, 2024. URL `https://openreview.net/forum?id=1tZbq88f27`.