

---

# YUAN3.0 FLASH: AN OPEN MULTIMODAL LARGE LANGUAGE MODEL FOR ENTERPRISE APPLICATIONS

---

YuanLab.ai  
research@yuanlab.ai

December 30, 2025

## ABSTRACT

We introduce Yuan3.0 Flash, an open-source Mixture-of-Experts (MoE) MultiModal Large Language Model featuring 3.7B activated parameters and 40B total parameters, specifically designed to enhance performance on enterprise-oriented tasks while maintaining competitive capabilities on general-purpose tasks. To address the overthinking phenomenon commonly observed in Large Reasoning Models (LRMs), we propose Reflection-aware Adaptive Policy Optimization (RAPO), a novel RL training algorithm that effectively regulates overthinking behaviors. In enterprise-oriented tasks such as retrieval-augmented generation (RAG), complex table understanding, and summarization, Yuan3.0 Flash consistently achieves superior performance. Moreover, it also demonstrates strong reasoning capabilities in domains such as mathematics, science, etc., attaining accuracy comparable to frontier model while requiring only approximately 1/4 to 1/2 of the average tokens. Yuan3.0 Flash has been fully open-sourced to facilitate further research and real-world deployment: <https://github.com/Yuanlab-LLM/Yuan3.0>.

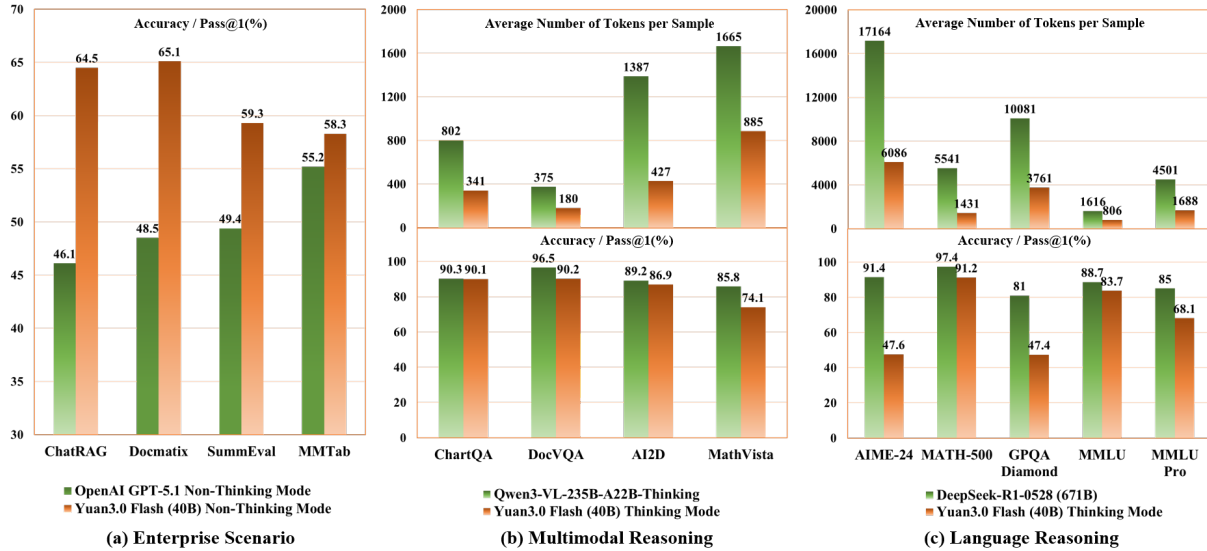


Figure 1: Performance-Efficiency Comparison of Yuan3.0 Flash across (a) Enterprise, (b) Multimodal, and (c) Language reasoning Benchmarks.

## 1 Introduction

Large language models (LLMs) have been applied to a wide range of tasks and domains [1, 2]. By integrating visual and linguistic domains, Multimodal Large language models (MLLMs) unite visual perception with natural language understanding, equipped with advanced comprehension and generation capabilities to tackle complex multimodal tasks [3, 4, 5, 6, 7]. Despite the rapid development of LLMs and MLLMs [6, 8], there are still shortcomings in key capabilities such as RAG, multimodal document processing, complex table analysis, summary generation, etc., when applying them to enterprise scenarios. These deficiencies directly limit their application in enterprise-level scenarios such as Intelligent Customer Service, R&D digitization, marketing analysis, etc.

Long chain-of-thought (CoT) reasoning has become a dominant research paradigm for both unimodal LLMs and MLLMs, driving significant advances in complex problem-solving [9, 8, 10]. The rapid evolution of MLLMs has extended the CoT paradigm to multimodal tasks, where models are required to reason over and integrate information from heterogeneous inputs like images, charts, and text [11, 12]. These models, by generating rationales that bridge visual perception and linguistic inference, have demonstrated remarkable capabilities in visual question answering, document analysis, and complex reasoning. However, this long CoT paradigm introduces the overthinking problem, where models generate excessively long reasoning traces, contributing to the waste of computational resources [13, 14].

In order to improve key capabilities in enterprise applications, and alleviate overthinking issue in LRMs, we carry out the following work:

- Develop a high-quality data generation system, which enables filtering of diverse multimodal data types, and generates high-quality pre-training and post-training datasets for vertical domains such as manufacture, finance, law, education, healthcare, etc.
- Propose a reflection-aware policy optimization (RAPO) reinforcement learning algorithm that effectively mitigates the overthinking issue of LRMs, and improves training efficiency and model accuracy.
- Train a multimodal model Yuan3.0 Flash with 40B parameters, which achieves superior performance on a wide range of enterprise-oriented benchmarks, and demonstrates strong capabilities in general tasks.

## 2 Model Architecture

The overall architecture of Yuan3.0 Flash consists of three components: a visual encoder, an MoE-based language backbone, and a lightweight multimodal alignment module as illustrated in Figure 2.

**MoE-based Language Backbone:** The language backbone of Yuan3.0 Flash employs a large-scale sparsely activated MoE architecture, aiming to effectively reduce computational costs while ensuring the model’s expressive capability. The architecture consists of 40 layers with 32 experts per layer, with two experts selected for forward computation using a Top-K routing strategy. The underlying architecture incorporates Localizing Filtering-based Attention (LFA) [15], introducing an explicit inductive bias into self-attention by favoring local token dependencies that are prevalent in natural language.

**MLP-based Vision-Language projector:** A lightweight MLP projector with SwiGLU as activation function is applied to align visual features to textual tokens, achieving stable and efficient visual-language alignment.

**Pretrained Vision Encoder:** We adopt a pretrained InternViT-300M [7] as vision encoder to facilitate deep visual reasoning and capture more fine-grained and complex visual features.

**Adaptive Image Segmentation Module:** To extract fine-grained detail from high-resolution images and reduce the computation costs, we propose an Adaptive Image Segmentation Module that automatically determines the optimal grid configuration for slicing an image. The detailed method is described in Appendix A.

## 3 Reinforcement Learning Methods

To address the overthinking issue and systematically advance reinforcement learning (RL) training for Yuan3.0 across diverse scenarios, we develop the Reflection-aware Adaptive Policy Optimization (RAPO) algorithm. By introducing the Reflection Inhibition Reward Mechanism (RIRM), RAPO effectively mitigates the common "overthinking" problem in LRMs. Furthermore, we improve the DAPO algorithm [16] by introducing adaptive training strategies and a unified reward system that supports multi-task RL training with a hybrid of thinking and non-thinking mode. These innovations significantly enhance Yuan3.0 Flash’s performance with stable and efficient large-scale RL training.

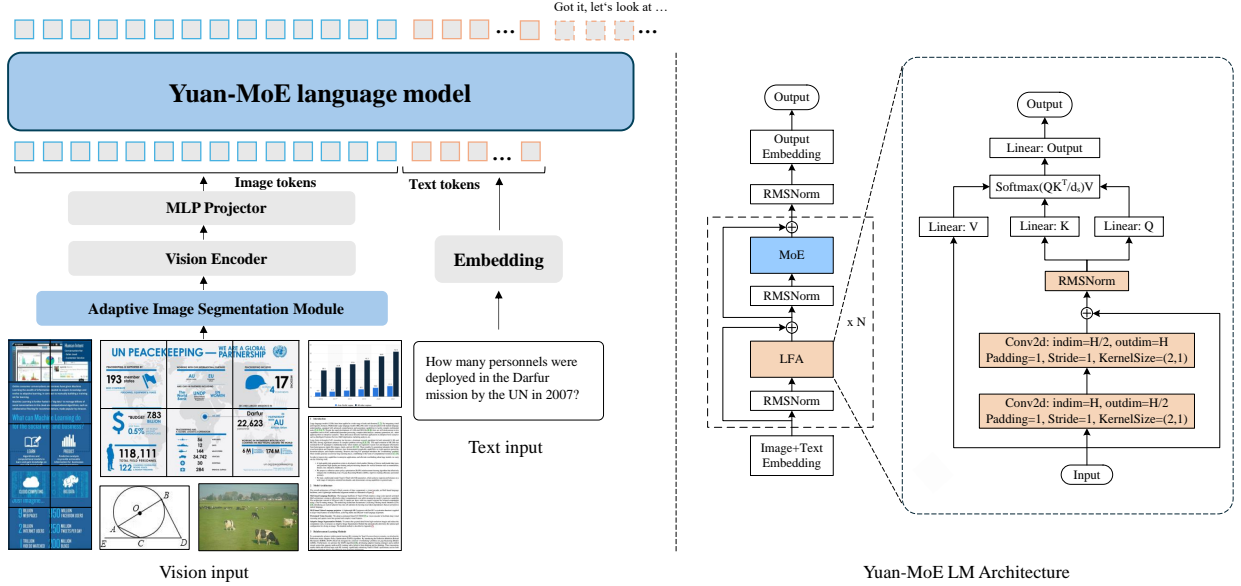


Figure 2: Overall architecture of Yuan3.0 Flash and MoE-based Language Backbone. The left figure depicts the proposed Yuan3.0 architecture, which comprises three integral components: (1) a ViT Encoder responsible for processing and encoding input images; (2) a lightweight MLP projector with SwiGLU activations to align visual features with the textual token space; and (3) a MoE-based Language Model that serves as the Language Decoder. The right figure presents the language backbone with Localizing Filtering-based Attention (LFA)

### 3.1 Reflection Inhibition Reward Mechanism (RIRM)

LRMs driven by reinforcement learning with verifiable rewards (RLVR) generate long reasoning chains to enhance inference capabilities. Models such as OpenAI’s o1 [10], DeepSeek-R1 [8], and Kimi-1.5 [17] have demonstrated the effectiveness of this approach. However, the RLVR mechanism tends to induce overthinking, which not only wastes computational resources but can also degrade model performance. This issue has become a critical bottleneck in practical LRM applications.

In complex reasoning tasks—particularly in mathematical and scientific domains—LRMs exhibit pronounced overthinking behavior. As illustrated in Figure 3(a), after correctly deriving an answer through a complete logical chain, models engage in repetitive self-checking and verification—a phenomenon termed "reflection". Our statistical analysis of Deepseek-R1 and DeepseekR1-Distill-1.5B [8] model on the AIME 2024 and MATH-500 benchmarks reveals that up to 71.6% of token consumption occurs in the post-answer reflection phase. (Figure 3(b)).

To mitigate overthinking problem, we propose the RIRM. RIRM identifies the step where the model initially arrives at the correct answer and its subsequent reflection steps during the reasoning process. It then assesses the reasoning trajectory along three key dimensions: whether the first correct answer is identified, whether the number of reflection steps is reasonable, and whether the final answer is correct. These dimensions are aggregated into a reward signal to guide policy optimization in reinforcement learning.

As shown in Figure 4, RIRM is implemented in three stages:

1. Extract reasoning trajectory from the model’s output, split them into equal-length strings, and use an LLM to annotate where the correct answer first appears and reflection steps with “[1<sup>st</sup>\_answer]” and “[verify]” respectively.
2. Calculate three scores using the annotated reasoning trajectory:

- $R_{ans}$ : whether the first correct answer is identified.
- $R_{ver}$ : whether the number of reflections,  $v$ , is reasonable. For  $R_{ver}$  calculation, samples with  $\geq 50\%$  accuracy rate use fixed bounds  $r_{min} = 2$ ,  $r_{max} = 10$ , while others use dynamic bounds  $r_{min} = \hat{r}_{min}$ ,  $r_{max} = \hat{r}_{max}$ .

**Question:**

Every morning Aya goes for a 9-kilometer-long walk and stops at a coffee shop afterwards. When she walks at a constant speed of  $s$  kilometers per hour, the walk takes her 4 hours, including  $t$  minutes spent in the coffee shop. When she walks at  $s+2$  kilometers per hour, the walk takes her 2 hours and 24 minutes, including  $t$  minutes spent in the coffee shop. Suppose Aya walks at  $s+\frac{1}{2}$  kilometers per hour. Find the number of minutes the walk takes her, including the  $t$  minutes spent in the coffee shop.

**Reasoning:**

<think>

Aya walks a 9-kilometer distance and stops at a coffee shop. When she walks at speed  $s$  km/h, the total time is 4 hours, including  $t$  minutes in the coffee shop. When she walks at  $s+2$  km/h, the total time is 2 hours 24 minutes, also including  $t$  minutes in the coffee shop. I need to find the total time when she walks at  $s + 1/2$  km/h, including the same  $t$  minutes in the coffee shop. First, ..... But the question asks for the number of minutes.  $9/3 = 3$  hours = 180 minutes.  $3 * 60 = 180$ , plus  $0.4 * 60 = 24$ , so 204 minutes.

Correct answer

What's that? But let me see the question: "find the number of minutes the walk takes her, including the  $t$  minutes." So 204 minutes. But let me confirm:  $s=2.5$  km/h,  $t=24$  min. New speed 3 km/h, walking time  $9/3=3$  hours = 180 min. Coffee 24 min, total 204 min.

Reflection #1

The problem gave  $s+1/2$ , which is 0.5, and I got  $s=2.5$ , so  $s+0.5=3$ , which is fine. But there is another way? The problem might expect the answer in minutes, and 204 seems straightforward. .... So inevitably I get  $s=2.5$ . Now for the new speed,  $s+0.5=3$  km/h, walking time  $9/3=3$  hours.  $3*60=180$  minutes, as before. Total time 3 hours + 24 minutes = 180 + 24 = 204 minutes.

Reflection #2

The problem says "the walk takes her", and includes the coffee time, so yes. But is the coffee time included in the walk? The phrase is "the walk takes her", but in the context, it says "the walk takes her 4 hours, including  $t$  minutes in coffee shop", so it's clear that "walk" here includes the entire trip, including the stop. So total time is 204 minutes.

Reflection #3

From the expression  $T = 4 + 9/(s+0.5) - 9/s$ ,  $9/s = 3.6$ ,  $9/(s+0.5) = 9/3 = 3$ . So  $T = 4 + 3 - 3.6 = 3.4$  hours. Same. With the quadratic, I got only one positive solution. The discriminant, I had 784, sqrt 28, etc.  $s=2.5$ .  $s+2=4.5$ .  $4.5/3=1.5$  hours = 90 minutes.  $s+1/2=3$  km/h.  $3/3=1$  hour = 60 minutes. It's correct. So the answer should be 204 minutes.

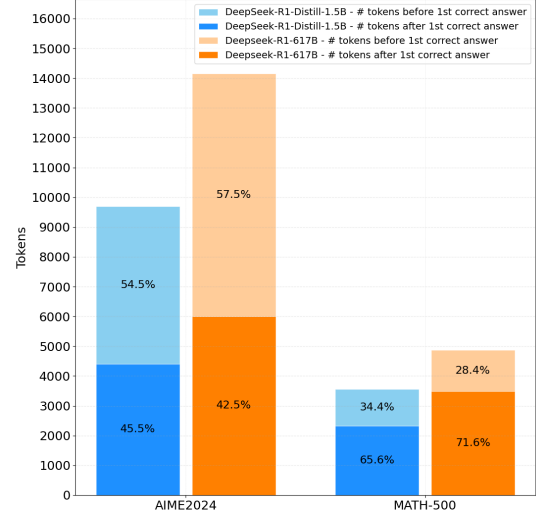
Reflection #7

</think>

**Summary:**

Aya walks a distance of 9 kilometers and stops at a coffee shop for  $t$  minutes each time. For speed  $s + \frac{1}{2}$  km/h, walking time =  $\frac{9}{s+0.5}$  hours = 3 hours = 180 minutes. Coffee shop time =  $t = 24$  minutes. Total time =  $180 + 24 = 204$  minutes. The total time, including the coffee shop stop, is 204 minutes.

(a)



(b)

Figure 3: (a) An example of Deepseek-R1’s reasoning process, showing repetitive "reflection" behavior after deriving the "first answer". (b) Average token consumption breakdown of DeepseekR1-Distill-1.5B and Deepseek-R1 across AIME 2024 and MATH-500 benchmarks. The light-colored segments indicate token consumption before the "first answer" appears, while the dark-colored segments correspond to token consumption during the "reflection" phase.

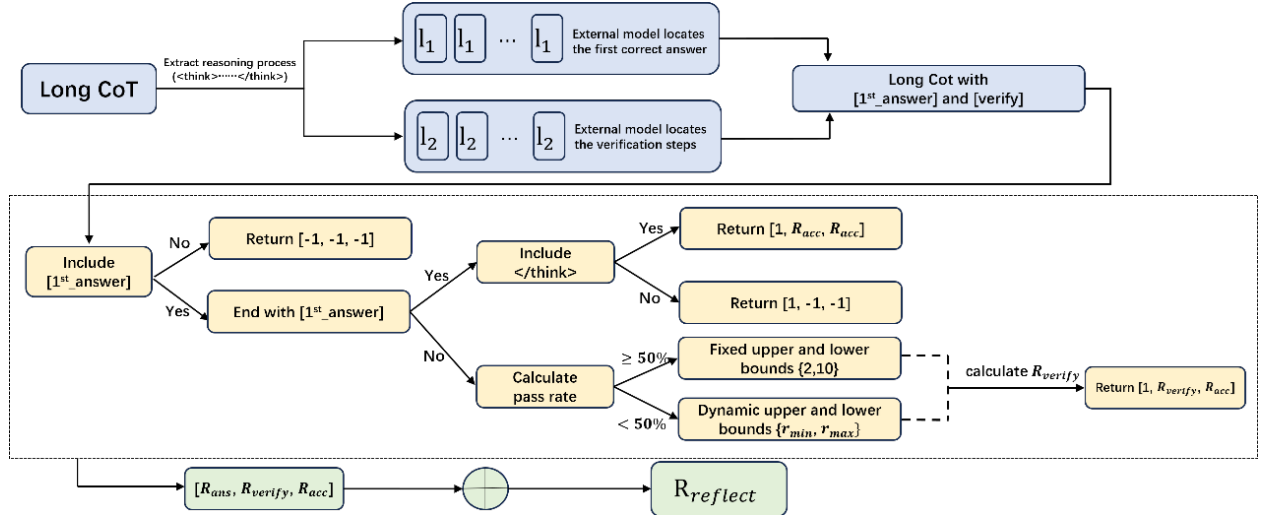


Figure 4: Illustration of Reflection Inhibition Reward Mechanism.

$\tilde{r}_{min}$  and  $\tilde{r}_{max}$  are the minimum and maximum number of "[verify]" among  $N$  samples of the same prompt.  $R_{ver}$  is calculated as:

$$R_{ver}(v) = \begin{cases} 1 & v \leq V_{min}, \\ 1 - \frac{(v - r_{min})}{(r_{max} - r_{min})} & r_{min} < v \leq r_{max}, \\ 0 & r_{max} < v. \end{cases} \quad (1)$$

- $R_{acc}$ : whether the final answer is correct.

3. The three component scores:  $[R_{ans}, R_{ver}, R_{acc}]$  are summed to obtain the final reflection inhibition reward:

$$R_{reflect} = R_{ans} + R_{ver} + R_{acc} \quad (2)$$

The incorporation of RIRM effectively mitigates overthinking issue and improve model accuracy for Yuan3.0 Flash. Table 1 presents a comparison of accuracy and output tokens on AIME 2024 and MATH-500 between the Yuan3.0 Flash 40B SFT base model trained with DAPO length penalty and with RIRM. Compared to the baseline model, the RIRM-based reinforcement learning framework yields a maximum accuracy improvement of 52.37% while simultaneously reducing output tokens by up to 47.14%. Figure 5 demonstrates the effects of RL training with RIRM, showing that the reasoning process of DeepseekR1-Distill-1.5B becomes significantly more concise. Notably, the overall token consumption is reduced by up to 64.09%, and the token consumption during the reflection phase see a reduction of up to 90.58%.

Table 1: Performance comparison of the Yuan3.0 Flash (40B) model under RL training with explicit DAPO length penalty versus the proposed RIRM.

Method	AIME 2024		MATH-500	
	Accuracy	Tokens	Accuracy	Tokens
Yuan3.0 Flash (40B) SFT	31.45	13,656	83.20	3,362
RL+DAPO length-penalty	46.35	13,781	89.06	3,974
RL+RIRM	<b>47.92</b>	<b>7,505</b>	<b>89.47</b>	<b>1,777</b>

**Question:**

Most people use C°(degree Celsius) to measure everyday temperatures. Astronomers prefer to use K (Kelvin) to measure temperatures. Approximately how many Kelvin are 25 C°? A.200 Kelvin B.250 Kelvin C.300 Kelvin D.350 Kelvin

**Reasoning:**

<think>

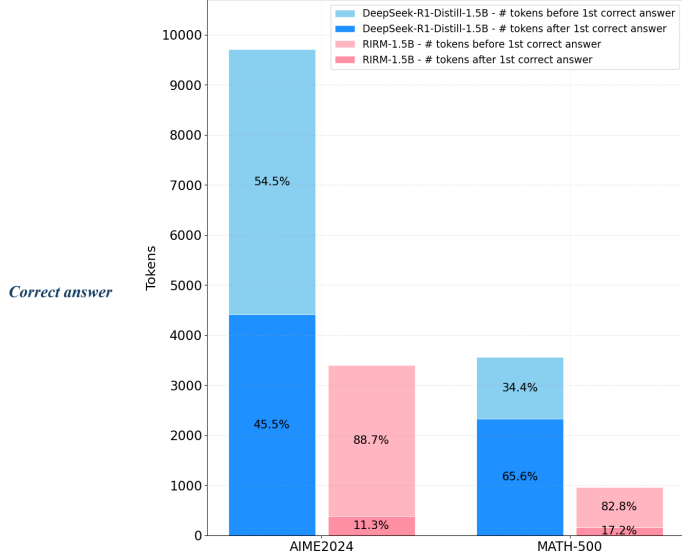
Okay, so the question is asking how many Kelvin 25 degrees Celsius is. Hmm, I remember that converting Celsius to Kelvin involves adding 273.15 to the Celsius temperature. Let me think. So, 25°C plus 273.15 equals... let me do the math. 25 plus 273 is 298, and then plus 0.15 makes it 298.15 K. So approximately 298 K. Looking at the options, the closest one is C. 300 Kelvin. Yeah, that makes sense. So the answer should be C.

</think>

**Summary:**

To convert 25°C to Kelvin, add 273.15. \n25°C + 273.15 = 298.15 K \n≈ 300 K.\n\boxed{C}<end\_of\_sentence><eod>

(a)



(b)

Figure 5: (a) Example of DeepseekR1-Distill-1.5B’s reasoning process after trained with RIRM. (b) Comparison of average token consumption of DeepseekR1-Distill-1.5B before and after trained with RIRM in AIME 2024 and MATH-500 benchmarks. The light-colored segments indicate token consumption before the "first answer" appears, while the dark-colored segments correspond to token consumption during the "reflection" phase.

### 3.2 Optimized DAPO

DAPO removes the KL penalty term and integrates key techniques—including the clip-higher mechanism, dynamic sampling, token-level policy gradient loss, and reward shaping for ultra-long sequences—to form a superior optimization

Table 2: DAPO enhanced with ADS substantially accelerates training of DeepSeek-R1-Distill-Qwen-1.5B compared to the baseline.

Case	Average time of generation (s)	Average time of step (s)
DAPO	455.73	928.48
DAPO+ADS	<b>257.22</b>	<b>437.18</b>

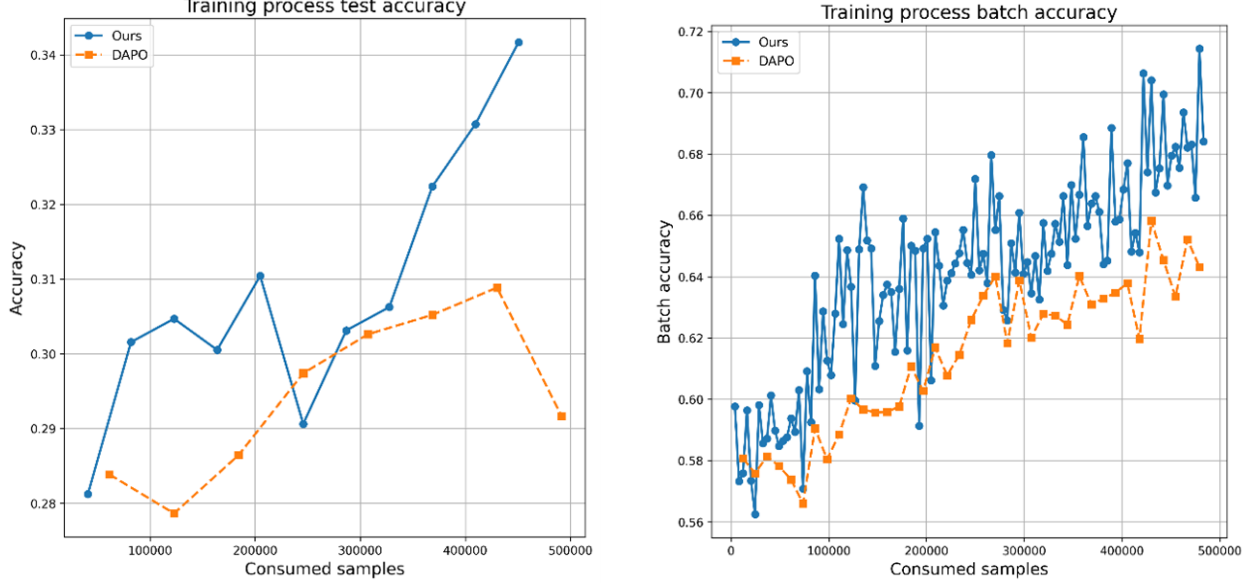


Figure 6: Training and testing accuracy of DeepSeek-R1-Distill-Qwen-1.5B under DAPO with and without ADS.

objective:

$$\begin{aligned}
 \mathcal{J}_{\text{DAPO}}(\theta) = & \mathbb{E}_{(q,a) \sim \mathcal{D}, \{o_i\}_{i=1}^G \sim \pi_{\theta_{old}}(\cdot|q)} \\
 & \left[ \frac{1}{\sum_{i=1}^G |o_i|} \sum_{i=1}^G \sum_{t=1}^{|o_i|} \min \left( r_{i,t}(\theta) \hat{A}_{i,t}, \text{clip}(r_{i,t}(\theta), 1 - \epsilon_{\text{low}}, 1 + \epsilon_{\text{high}}) \hat{A}_{i,t} \right) \right], \\
 & \text{s.t. } 0 < |\{o_i | \text{is\_equivalent}(a, o_i)\}| < G
 \end{aligned} \tag{3}$$

To further enhance computational performance and stabilize gradient updates, we have introduced a series of optimizations to the DAPO algorithm.

**Adaptive Dynamic Sampling** DAPO’s dynamic sampling method oversamples and filters out prompts with identical reward scores, to ensure all prompts in a batch have valid gradients while maintaining a consistent prompt count. However, it typically requires sampling three times the training batch size to guarantee sufficient valid samples, with excess samples being discarded. This leads to data waste, increases training iteration time, and increase the need for additional training epochs.

To improve sampling efficiency, we develop the Adaptive Dynamic Sampling (ADS) strategy that refines DAPO’s dynamic sampling process. We first sample a batch matching the training batch size and filter out prompts with identical rewards. For the remaining prompts, we compute their passing rates:  $P_j = \frac{1}{G} \sum_{i=1}^G I(a_i = 1)$ . We then sort prompts by passing rate in descending order and resample from the highest-pass-rate prompts to replenish the batch. To control resampling intensity, we replenish the batch to the smallest multiple of the mini-batch size, ensuring sufficient samples for training without excessive duplication. For detailed algorithm specifications, please refer to the Appendix B. As shown in Table 2, DAPO with ADS achieves a significant reduction in both average sampling time and per-step iteration time compared to the baseline DAPO, resulting in a 52.91% improvement in training efficiency. Correspondingly, Figure 6 illustrates the model’s consistent accuracy improvement throughout both training and testing phases.

**80/20 rule** DAPO encounters training instability issues when applied to large-scale Mixture-of-Experts (MoE) models. Building upon DAPO, we apply the 80/20 algorithm [18], which updates gradients using only the top 20% of tokens with the highest entropy during RL training. This approach not only significantly enhances training stability but also improves model accuracy.

**Optimized dual-clip** During large-scale RL training, the updated policy model may significantly deviate from the old policy, resulting in abnormally large probability ratios between new and old policies for certain trajectory data. When the advantage values are negative, these extreme ratios can lead to unbounded gradients, causing model collapse. While dual-clip PPO [19] is commonly used to mitigate such instability, it introduces an additional hyperparameter  $c$ . Improper tuning of  $c$  may hinder model convergence, and the extra bounds checking on the product of probability ratios and advantage values increases computational overhead as training data scales up. To address this, we optimize the standard PPO’s clipped probability ratio approach: when encountering abnormally large probability ratios with negative advantages, we exclusively use the clipped ratio to prevent extreme gradient values, thereby further stabilizing gradient updates.

Integrating the aforementioned optimizations, we compute the RL loss using the following formula:

$$J^{\mathcal{B}_r} = \mathbb{E}_{\mathcal{B}_r \sim \mathcal{D}, (q, a) \sim \mathcal{B}_r, \{o_i\}_{i=1}^G \sim \pi_{\theta_{old}}(\cdot|q)} \left[ \frac{1}{\sum_{i=1}^G |o_i|} \sum_{i=1}^G \sum_{t=1}^{|o_i|} \mathbb{I}[H_t^i \geq \tau_\rho] PG_{i,t,\epsilon_{low},\epsilon_{high}}(\theta) \right], \text{ s.t. } |\mathcal{B}_r| = k \cdot |\mathcal{B}_{mini}| \quad (4)$$

where

$$PG_{i,t,\epsilon_{low},\epsilon_{high}}(\theta) = \begin{cases} \text{clip}(r_{i,t}(\theta), 1 - \epsilon_{low}, 1 + \epsilon_{high}) \hat{A}_{i,t}, & \text{if } \hat{A}_{i,t} \leq 0, \\ \min(r_{i,t}(\theta) \hat{A}_{i,t}, \text{clip}(r_{i,t}(\theta), 1 - \epsilon_{low}, 1 + \epsilon_{high}) \hat{A}_{i,t}) & o.w. \end{cases} \quad (5)$$

In the equations, red text highlights our algorithmic optimizations.  $H_i^t$  denotes the entropy of token  $t$  in response  $i$ .  $\mathbb{I}[\cdot]$  is the indicator function, which takes the value 1 if the condition holds and 0 otherwise.  $\rho = 0.2$  is the top proportion of high-entropy tokens we specify for each response.  $\tau_\rho$  is the corresponding entropy threshold, such that only tokens with  $H_i^t \geq \tau_\rho$  are used for gradient computation; these tokens constitute the top  $\rho$  portion of all tokens in the response.  $\mathcal{B}_{mini}$  denotes the mini-batch size, and after adaptive dynamic sampling, the global batch size  $\mathcal{B}_r$  becomes  $k$  times the mini-batch size.

The optimized DAPO algorithm (Figure 7) maintains the lowest gradient norm, indicating high training stability. This contrasts with DAPO without the 80/20 rule, whose gradient norm oscillates and sharply increases in later stages, leading to rapid collapse. The stability of our approach is also evidenced by moderate length growth and a gradual entropy increase, a pattern correlated with better model performance as suggested in [16].

### 3.3 Multi-Task Training with Adaptive Strategies

Unlike Deepseek-R1 or Qwen 3 series models, which strictly separate deep thinking training from fast thinking training, or Kimi1.5 which trains two separate models for these distinct capabilities, we adopt a **mixed training paradigm** that jointly trains a single model on both thinking tasks (including language and multimodal mathematical reasoning, code generation, and scientific tasks) and non-thinking tasks (such as language and multimodal RAG, language and multimodal chat, and general scenario QA). This unified approach thus enables the model to achieve superior generalization and adaptability across a wide range of domains.

To preserve task-specific characteristics, we employ distinct training strategies:

- For deep thinking tasks involving language/multimodal mathematical reasoning and scientific tasks, we utilize RIRM.
- For code generation under thinking tasks and all non-thinking tasks, we apply optimized DAPO with explicit length constraints and penalties

Different maximum output limits and penalty intervals are carefully configured for each task category to prevent interference and ensure training stability.

**Data grouping and alternating training strategy** When data from different tasks have widely varying maximum output lengths (e.g., 4K vs. 16K), traditional mixed batching would force shorter sequences to idle while waiting for the longest one in a batch, severely hampering efficiency. To address this, we introduce a data grouping and alternating



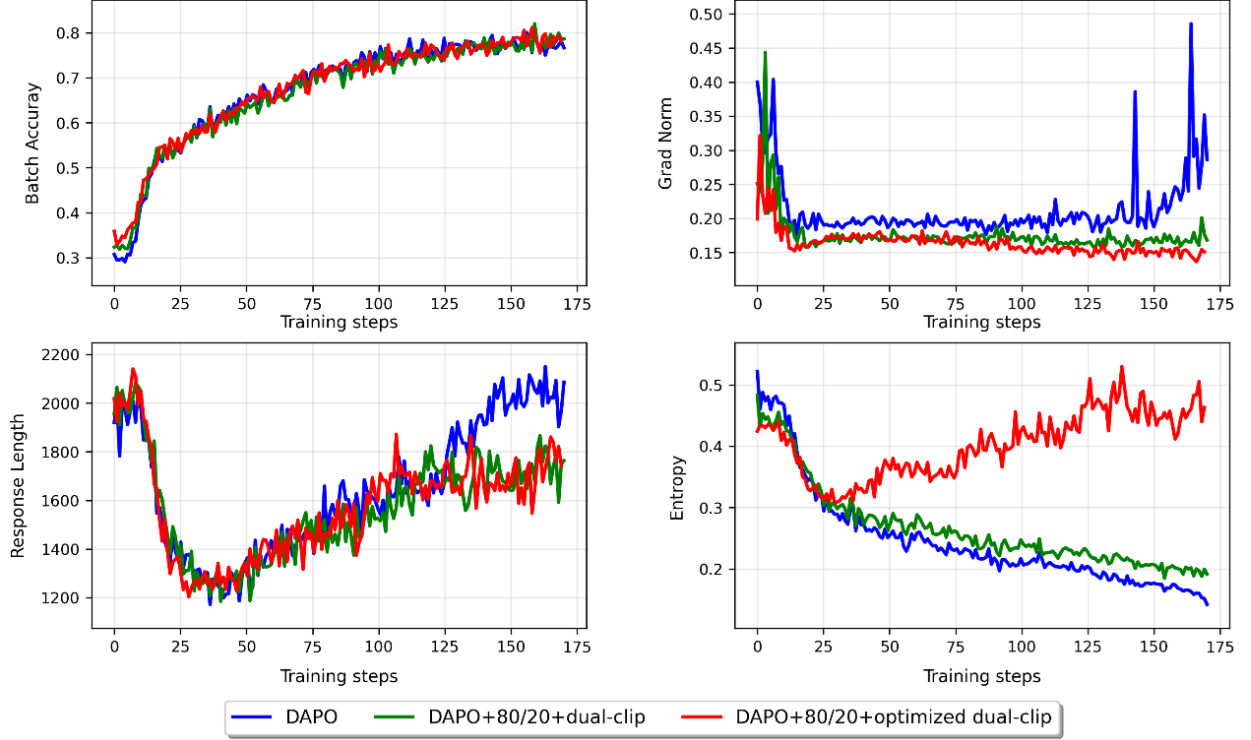


Figure 7: Training dynamics of DAPO (blue line), DAPO with 80/20 rule and dual-clip technique (green line), and DAPO with 80/20 rule and our optimized dual-clip technique.

training strategy based on maximum output lengths: data are first grouped by length categories (e.g., 4K group, 16K group), and then trained alternately according to their proportion (e.g., with a 4K:16K ratio of 2:1, two 4K batches are followed by one 16K batch). Experiments on the Yuan3.0 Flash 40B model show that this method improves inference phase throughput by 16%.

**Repetition truncation** Furthermore, to maintain output quality during reinforcement learning, we implement repetition truncation in trajectory collection. The model’s output is monitored in real time, and every 1024 tokens generated undergo repetition detection. If any continuous subsequence of length  $N$  repeats beyond a preset threshold, the episode is terminated early to prevent degenerate output loops. In our setup,  $N$  and the repetition threshold are set to 200 and 10, respectively.

### 3.4 Unified Reward System: Verifiable Framework and Generative Reward Models

To comprehensively guide model optimization across both objective and open-ended tasks, we establish a Unified Reward System consisting of two complementary components: the Verifiable Reward Framework for tasks with deterministic criteria, and Generative Reward Models (GRMs) for open-ended scenarios requiring nuanced human-like judgment.

**Verifiable Reward Framework** The framework provides precise, rule-based evaluation across specialized domains:

*Mathematics & Science.* For mathematical reasoning and scientific tasks (covering set operations, chart interpretation, and multiple-choice questions), we employ a hybrid evaluation approach: rule-based string matching for preliminary scoring, augmented by the "math\_verify" technique to rigorously validate the equivalence of mathematical expressions and formula-based answers.

*Programming Languages:*

- **Python:** Our automated testing framework executes unit tests under multiple scenarios: standard input simulation, direct code execution, and text-embedded code extraction. Leveraging Uvicorn for asynchronous execution enables high-throughput evaluation.



- Generated statements are executed against database engines, with correctness determined by result consistency with reference queries.

*Retrieval-Augmented Generation (RAG).* The RAG evaluation system employs a multi-dimensional strategy combining:

- Objective metrics: Threshold judgments, linear mapping, and exact matching for factual consistency
- Semantic evaluation: Instruction adherence and semantic understanding via LLM discriminators for quality ranking and classification
- Text similarity: Normalized mapping of similarity metrics for generation quality assessment

*Output Quality Constraints.* To enforce standardization, we implement a multi-faceted negative reward mechanism:

- Formatting Control: Regex-based validation of thought-process tags ("`<think>...</think>`") and structural markers ("`<|Assistant|>`", "`<|end_of_sentence|>`"), with -1.0 penalty for deviations
- Language Consistency: Character-level analysis of Chinese/English ratios (post number/symbol filtering), applying -1.0 penalty when non-target language exceeds 20%
- Repetition Penalty: N-gram sliding window detection of token repetition patterns, triggering -1.0 penalty when unique character falls below 80%.

**Generative Reward Models (GRMs)** To address the lack of definitive answers in open-ended and subjective tasks, we employ Generative Reward Models (GRMs) for task-specific scoring. Compared to scalar-based and semi-scalar approaches, GRMs offer superior interpretability and generalization by translating advanced language understanding and visual perception into quantifiable rewards [20]. We train both pure-language and multimodal GRMs using Formatted Supervised Fine-Tuning (SFT) and Rejection Sampling Fine-Tuning (RFT), transforming the model’s comprehension into rigorous judging capabilities through instance-specific scoring guidelines and iterative refinement.

*Pure-Language GRM.* We curate approximately 1M seed instruction-tuning samples spanning multi-turn dialogue, safety, open-ended Q&A, riddles, text processing, and social sciences (politics, history, religion, literature, art, economics). For each query, we generate 7 diverse responses using Yuan3.0 Flash 40B, mixing them with SFT answers to form candidate lists. A teacher model evaluates these responses based on predefined standards, reasoning, and a 1–10 score. To ensure reward accuracy, we apply ground-truth consistency filtering—only retaining samples where the original answer uniquely achieves the highest score. The rejection sampling phase further enhances precision by incorporating error-marked data, outputs from larger models, and open-source preference datasets (UltraFeedback [21], OffsetBias [22], Skywork-Reward [23], HelpSteer2 [24], JudgeLM [25]). The fine-tuned reward model performs three inferences per sample, with ground truth filtering reapplied to retain high-quality data for final training.

*Multimodal GRM (MGRM).* MGRM extends evaluation to cross-modal semantic alignment, focusing on image content, chart logic, and on-screen information. Optimized for tasks including image captioning, chart analysis, OCR, code reasoning, and screenshot analysis, MGRM trains on a combination of internal data and open-source preference datasets (RLAIF-V [26], RLHF-V [27], POVID [28], MM-RLHF [29], VL-Feedback [30], wildvision-battle [31]). A multimodal teacher model generates formatted scoring data to enhance the model’s assessment of semantic accuracy, logical correctness, and descriptive clarity. RFT further improves MGRM’s adaptation to larger model distributions and data diversity. The fine-tuned model performs secondary inference and filtering on error data, with targeted debiasing (hinted sampling) ensuring stable, human-aligned feedback within the RL loop.

## 4 Datasets

### 4.1 Pre-training Data

A mixed corpus of text and image-text pairs was built to train the Yuan3.0 Base model.

#### 4.1.1 Textual Datasets

The pre-training textual data with 3.5TB tokens is sourced from diverse origins - including web pages, encyclopedia entries, book excerpts, academic papers and code. To guide the curation of web-crawled data, we first train a FastText-based domain classifier using a labeled domain dataset. This classifier is capable of distinguishing across a broad spectrum of domains, including advertising, economy, healthcare, law, business, etc. Through sampling analysis of open-source web-crawled datasets [32] by our domain classifier, it is indicated that advertisements, lifestyle news, and entertainment/sports news accounts for the majority (Figure 8).

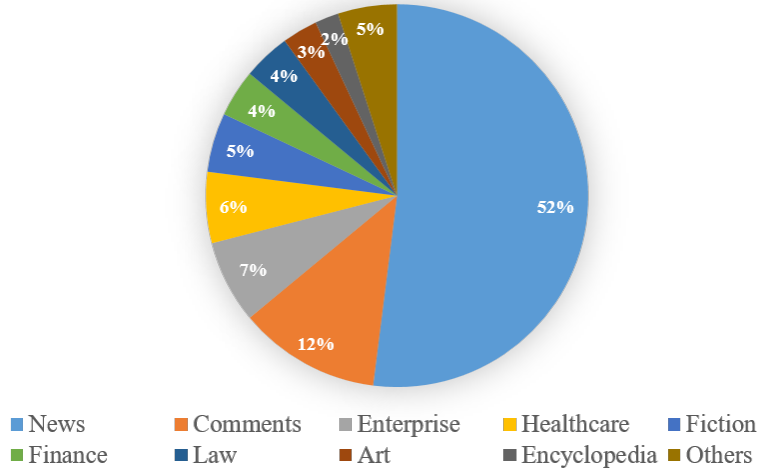


Figure 8: Domain-wise composition of the web-crawled pre-training corpus.

We reduce the proportion of low-value news and advertising content while keeping high-value data, such as technology, science, knowledge, etc. Furthermore, we increase the proportion of data from vertical domains, such as finance, law, manufacturing, healthcare, etc. We also train a quality-scoring classifier and remove all text samples with a normalized quality score below 0.6 (on a 0–1 scale) to ensure data quality.

#### 4.1.2 Multi-modal Datasets

The multimodal pre-training of Yuan3.0 Base is designed to establish robust visual-linguistic alignment and reasoning capabilities. We construct a large-scale, diverse dataset of 1.5B image-text pairs. This comprehensive corpus enables the model to learn deep semantic relationships between visual content and textual descriptions, forming a foundation for sophisticated, human-like multimodal reasoning.

### 4.2 Fine-tuning datasets

#### 4.2.1 Datasets for General tasks

For general dialogue tasks such as casual chat as well as domain knowledge QA, a dedicated scoring model is trained to construct a multi-dimensional evaluation system focusing on the sample quality. The model conducts comprehensive evaluation focusing on four key dimensions: verifying whether the answer accurately matches the user’s question without deviating from the topic or being irrelevant, assessing the coherence and organization of the answer to ensure clear reasoning and logical closure, ensuring responses comply with the inherent constraints strictly removing vulgar, non-compliant, and misleading content. Only samples with scores passing the preset threshold are retained. The multi-modal dataset spanning multiple domains, including documents, charts, visual reasoning problems, code, OCR, science problems, etc., is constructed.

#### 4.2.2 Datasets for Enterprise Scenario

We construct a hybrid dataset highly relevant to enterprise application scenarios, including RAG, tabular, etc., by integrating human annotations, synthetic generation, and open-source resources. We create a 45K-dialogues dataset (25K text-only, 20K multimodal) by human expert to keep multimodal samples aligned with visual inputs including code snippets, diagrams, API screenshots, etc. We generate synthetic data across wide range of domains including Literature, Geography, Biology, Economics, etc., leveraging data sources such as Wikipedia, arXiv, etc. The synthetic data includes 305K samples (160K text-only, 145K multimodal), which ensures semantic diversity, covers 95% of low-frequency concepts. We also build general instruction datasets [33, 34] and multimodal understanding datasets [35, 36, 37, 38, 39, 40, 41] covering 200K text samples and 160K multimodal samples.

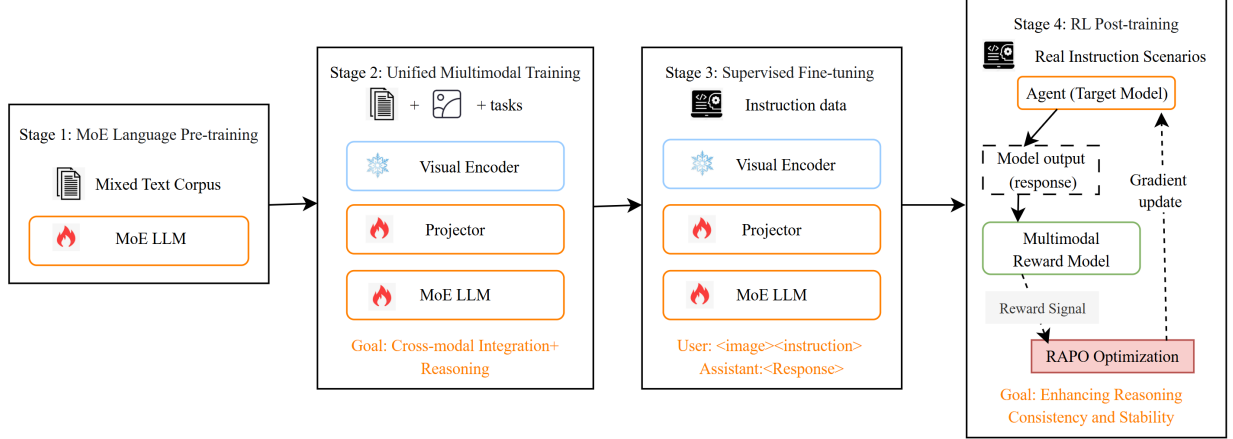


Figure 9: Structure of multi-stage collaborative multimodal training framework.

### 4.3 Datasets for Reinforcement Learning

We have designed a standardized workflow to generate reinforcement learning datasets for a wide range of domains. Firstly, data with ground truth answers are extracted as candidate data. For code data, the answer comprises several unit test cases (with more than 5 unit tests). Secondly, for mathematical and scientific data, a rule-based filtering is applied to remove samples including multiple choice and proof-based questions. Next, the remaining samples are graded by the pass rate of Yuan3.0 Flash SFT model. Only those difficult samples are retained for subsequent reinforcement learning training.

## 5 Training Pipeline

The study constructs a unified, multi-stage collaborative multimodal training framework, designed to systematically achieve the alignment of visual and linguistic semantic spaces, the integration of cross-modal knowledge, and the enhancement of complex multimodal reasoning abilities in a stepwise manner. The overall process comprises four stages (Figure 9). First, the Yuan3.0 language model is pre-trained from scratch under a unified large-scale pre-training pipeline on about 3 trillion tokens. Second, the MoE language backbone and projector are unfrozen to perform unified multimodal training on 256 million image-text pairs. The third stage involves supervised fine-tuning based on high-quality instruction data, further improving the model’s multimodal instruction-following and reasoning capabilities. Finally, large-scale reinforcement learning is conducted, continuously optimizing the model’s reasoning consistency and behavior quality in real interaction scenarios. The RL training adopted a hybrid training mode integrating thinking and non-thinking capabilities.

## 6 Experiments

### 6.1 Long-Text Benchmarks

Yuan3.0 supports an extended context window of 128K tokens, enabling it to seamlessly process and reason over lengthy, complex documents. To assess the model’s ability on processing and understanding long textual data, we evaluate Yuan 3.0 Flash model using the Needle-in-a-Haystack (NIAH) benchmark [42]. The experiments are conducted across a range of context lengths and needle positions, with response accuracy serving as the evaluation metric. The results are presented in Figure 10. It can be concluded that our model achieves a perfect score within the 128K context scope. This indicates that the model can stably and accurately retrieve critical target information from ultra-long text corpora, which provides a reliable technical guarantee for its deployment in enterprise-level long-text application scenarios (e.g., long report analysis, multi-chapter knowledge reasoning).

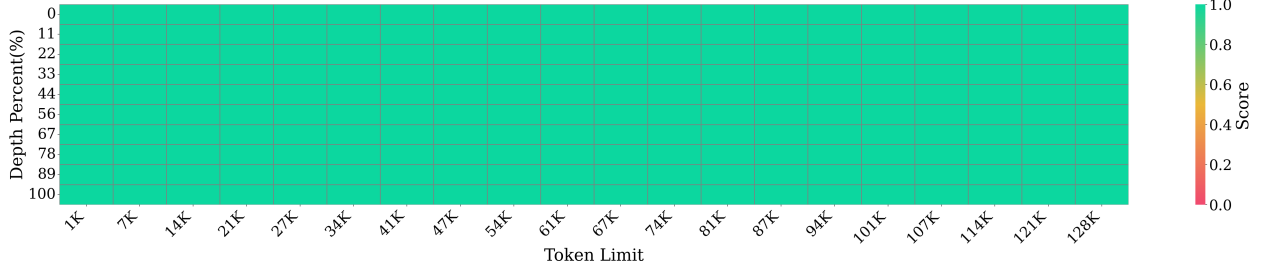


Figure 10: Evaluation results by NIAH test.

Table 3: Comparison among Yuan3.0 Flash and other representative models on Docmatix

Model	Acc(%)
Qwen2.5-VL-72B-Instruct	59.75
InternVL3-78B	42.99
Claude3.5-Sonnet	42.55
OpenAI GPT-4o	56.79
OpenAI GPT-o3	45.57
OpenAI GPT-5.1	48.52
Yuan3.0 Flash	65.07

## 6.2 Enterprise scenarios benchmarks

We further evaluate Yuan3.0 Flash on a suite of complex, enterprise-oriented tasks, covering multimodal retrieval, text retrieval, complex table understanding, text summarization, and tool invocation, which together reflect representative real-world application requirements.

For multimodal retrieval, we adopt the Docmatix benchmark [43] for multimodal RAG evaluation, which assesses a model’s ability to retrieve, associate, and accurately answer questions across multiple modalities (text, tables, and images) within multi-page, complex documents. On this benchmark, Yuan3.0 Flash achieves the highest performance among all of the models such as Claude 3.5, GPT-4o, and GPT-5.1 as shown in Table 3. This result demonstrates strong multimodal reasoning and retrieval capabilities in document-level scenarios.

For text retrieval, we evaluate on ChatRAG [44], a widely recognized industrial benchmark comprising ten tasks. These tasks include long-context retrieval such as D2D, QuAC, QReCC, short-context and structured retrieval including CoQA, DoQA, CFQA, SQA, HDial, as well as Wikipedia-based retrieval tasks including TCQA, INSCIT. Across the full ChatRAG task set, Yuan3.0 Flash attains an average accuracy of 64.47, surpassing DeepSeek-V3, DeepSeek-R1, GPT-4o, and GPT-o3, and achieves leading performance on 9 out of the 10 tasks, indicating robust retrieval effectiveness across diverse textual contexts as shown in Table 4.

Multimodal table understanding is a critical capability in enterprise office scenario. We evaluate this ability using MMTab [45], which consists of 15 authoritative benchmarks spanning multiple task categories. Firstly, question-answering tasks include the following benchmarks. TABMWP contains multiple table-based math word problems. WTQ and HiTab are the tasks of complex querying and reasoning over Wikipedia tables. TAT-QA is a benchmark for joint reasoning over financial tables and text. FeTaQA is a benchmarks for generating free-form answer from tables. Secondly, Fact-checking task includes TabFact, InfoTabs, HiTab T2T, Rotowire, and WikiBIO. Then, TSD, TCE, TCL,

Table 4: Comparison among Yuan3.0 Flash and other representative models on ChatRAG (Acc. %)

Models	Avg All	D2D	QuAC	QReCC	CoQA	DoQA	CFQA	SQA	TCQA	HDial	INSCIT
DeepSeek-V3	50.47	31.59	28.86	49.31	76.98	26.11	83.49	82.13	46.69	47.43	32.08
OpenAI GPT-4o	50.54	32.76	26.56	49.30	76.11	28.78	81.85	81.14	49.75	41.29	26.69
OpenAI GPT-o3	44.06	23.05	20.82	40.42	69.42	18.56	67.75	86.71	45.85	41.29	26.69
DeepSeek-R1	39.42	21.46	22.23	42.41	62.53	24.68	81.48	82.06	30.74	37.97	28.68
OpenAI GPT-5.1	46.10	28.24	23.16	45.43	68.84	20.88	73.05	81.32	44.70	45.39	29.95
Yuan3.0 Flash	<b>64.47</b>	<b>49.82</b>	<b>53.79</b>	<b>57.08</b>	<b>90.93</b>	<b>59.99</b>	74.40	<b>87.52</b>	<b>66.31</b>	<b>68.45</b>	<b>36.40</b>

MCD, and RCE focus on long-context table-centric information processing. As shown in Table 5, Yuan3.0 Flash achieves a leading average accuracy of 58.29, exceeding GPT-5.1, and outperforms competing models on 7 of the 15 tasks, demonstrating comprehensive and well-balanced table reasoning and generation capabilities.

Table 5: Comparison among Yuan3.0 Flash and other representative models on MMTab

Tasks		GLM-4.5V	OpenAI GPT-4V	OpenAI GPT-5.1	Yuan3.0 Flash
<b>Avg.</b>		52.00	29.90	55.15	<b>58.29</b>
<b>QA</b>	TABMWP (Acc.%)	88.21	60.50	64.95	<b>95.09</b>
	WTQ (Acc.%)	77.42	48.00	60.77	68.23
	HiTab (Acc.%)	51.52	27.50	77.77	69.80
	TAT QA (Acc.%)	62.69	32.50	61.37	<b>69.17</b>
	FeTaQA (BLEU)	5.25	11.04	8.70	<b>28.42</b>
<b>Fact Checking</b>	TabFact (Acc.%)	89.44	45.50	52.81	87.32
	InfoTabs (Acc.%)	79.48	65.60	64.30	<b>83.50</b>
	HiTab T2T (BLEU)	5.17	2.98	44.16	13.30
	Rotowire (BLEU)	4.48	4.23	17.81	14.74
	WikiBIO (BLEU)	2.69	1.94	11.95	<b>17.26</b>
<b>TSD</b>	Row (Acc.%)	47.40	19.00	96.60	46.60
	Col (Acc.%)	89.70	38.00	62.10	82.80
<b>TCE (Acc.%)</b>		52.74	14.36	86.43	56.77
<b>TCL (Acc.%)</b>		50.84	27.91	44.66	<b>56.98</b>
<b>MCD (F1%)</b>		43.47	3.50	72.46	65.20
<b>RCE</b>	Row (F1%)	50.77	48.52	53.58	<b>62.07</b>
	Col (Acc.%)	82.79	57.14	57.20	73.67

For text summarization, which is essential for compressing user history in agent-based applications, we evaluate on SummEval [46], covering lexical overlap, semantic similarity, and factual consistency. Yuan3.0 Flash achieves an average score of 59.31 as shown in Table 6, outperforming Gemini (45.35) and GPT-5.1 (49.44), highlighting its strength in producing concise, semantically faithful, and factually consistent summaries.

Finally, tool invocation ability is assessed using BFCL V3 [47], which evaluates real-world function-calling competence across five tasks as shown in Table 7. Firstly, Non-Live AST evaluates the ability of static function selection and argument extraction. Secondly, Live AST is a benchmark evaluating the ability of dynamic execution with real-time feedback. Thirdly, Multi-turn demonstrates the ability of context maintenance and multi-tool coordination. Last, Relevance Detection evaluates deciding whether tool invocation is required, and Irrelevance Detection (rejecting invalid or unnecessary calls). Yuan3.0 Flash demonstrates consistently strong performance across all categories (57.97 on average) with no evident weaknesses, underscoring its reliability and maturity in tool-augmented reasoning and execution.

Overall, these results indicate that Yuan3.0 Flash delivers competitive and often leading performance across a broad spectrum of enterprise-level complex tasks, with particular strengths in multimodal retrieval, table understanding, summarization quality, and robust tool-calling behavior.

### 6.3 Multimodal Benchmarks

To comprehensively evaluate the capabilities of Yuan3.0 Flash model across diverse visual-language understanding scenarios, we evaluate on the following benchmarks. AI2D [48] is a benchmark for reasoning over science diagrams, requiring models to parse visual elements and their relationships to answer questions. ChartQA [49] evaluates chart comprehension by requiring models to extract and reason about data from various chart types (e.g., bar, line, pie) to answer factual and reasoning questions. DocVQA [50] tests the understanding of scanned documents, requiring models to answer questions by interpreting textual content, layout, and visual cues within document images. MathVista [51] is a benchmark designed to evaluate mathematical reasoning in visual contexts, combining problems from various existing datasets that require jointly understanding figures, plots, and diagrams to perform mathematical calculation and reasoning.

As shown in Table 8a, we evaluate models in non-thinking mode across four tasks. For ChartQA and DocVQA, YUAN3.0 Flash performs close to Qwen3-VL-32B and Qwen2.5-VL-72B, demonstrating solid abilities on OCR and document scenarios. For AI2D, the performance of Yuan3.0 Flash is slightly behind other models but has comparable

Table 6: Comparison among Yuan3.0 and other representative models on SummEval.

Models	Avg. All (W=100%)	Word Overlap		Semantic Similarity BERTScore (F1%)	Factual Consistency SummaC (Acc. %)
		ROUGE-1 (F1%)	ROUGE-2 (F1%)		
DeepSeek-V3	59.28	25.50	9.20	86.30	68.20
DeepSeek-V3.2	51.36	33.30	11.92	85.61	41.76
Gemini-2.0-Flash	45.35	24.80	8.70	85.70	29.50
Claude-3.5-Sonnet	45.43	24.10	8.30	85.20	30.70
OpenAI GPT-4o	46.53	25.00	8.90	85.90	32.50
OpenAI GPT-5.1	49.44	27.48	10.16	84.63	40.50
Yuan3.0 Flash	<b>59.31</b>	<b>51.32</b>	<b>28.32</b>	<b>89.99</b>	45.34

Table 7: Comparison among Yuan3.0 Flash and other representative models on BFCL.

Model	Avg. All	Non-Live AST	Live AST	Multi Turn	Relevance Detection	Irrelevance Detection
Qwen3-235B-A22B	67.94	87.90	77.03	40.12	83.32	76.32
Claude-3.7-Sonnet	58.58	41.29	78.41	48.38	72.22	81.40
OpenAI GPT-4.1-nano	52.98	76.65	64.33	19.88	94.44	59.39
OpenAI o3-mini	51.26	42.12	77.30	26.12	77.78	80.67
Llama-4-Scout-17B-16E-Instruct	45.41	83.48	57.97	1.88	100.00	39.66
Yuan3.0 Flash	57.97	69.06	66.37	36.88	94.44	77.09

scores, reflecting the ability of the model in handling structured and scientific reasoning tasks. In MathVista task, Yuan3.0 Flash scored 72.5, which is close to Qwen2.5-VL-72B but behind Qwen3-VL-32B. The comparisons prove the capabilities in mathematical computation and logical reasoning. As for thinking mode, the performance of Yuan3.0 Flash Thinking Mode showed gains in some tasks compared to its non-thinking mode. Comparing with other models, Yuan3.0-40B has higher performance than Qwen3-VL-32B-thinking and is close to Qwen3-VL-235B-A22B on ChartQA as shown in Table 8b. For AI2D task, Yuan3.0 Flash achieves a slight increase in non-thinking mode and comes close to Qwen3-VL-235B-A22B and Qwen3-VL-32B. In the MathVista task, Yuan3.0 Flash obtained a score higher than the non-thinking score, but has a lower score than Qwen3-VL-235B-A22B and Qwen3-VL-32B. To further evaluate and compare the computational efficiency of the reasoning process and its trade-off with accuracy, we collected statistics on the average number of output tokens generated during inference for our model and other baseline models. As shown in Table 8c, Yuan3.0 Flash Thinking Mode significantly reduces computational cost compared to the two Qwen3 models. Specifically, its token count represents a reduction to between 1/3 and 1/2 of that of Qwen3-VL-235B-A22B, effectively mitigating overthinking issue and reducing unnecessary resource consumption in the reasoning process.

In summary, the minimal performance gap between its two modes of YUAN3.0 Flash highlights the practical efficiency of the lightweight non-thinking design. In addition, Yuan3.0 Flash Thinking Mode maintains accuracy comparable to Qwen3 series models while significantly reduces the token consumption.

## 6.4 Evaluation on General Reasoning Benchmarks

To evaluate mathematical, coding, etc., reasoning capability, we conduct benchmarks on MATH-500 [52], AIME 2024 [53], LiveCodeBench [54], HumanEval [55], MMLU, MMLU-Pro[56], and GPQA-Diamond [57]. We averaged 64 samples per question for AIME, while 8 samples per question for others.

The Table 9 presents comparison of the Yuan3.0 Flash model with Deepseek-V3-0324 under non-thinking mode. Yuan3.0 Flash (40B) demonstrates performance gaps relative to DeepSeek-V3-0324 (671B) on AIME2024, GPQA-Diamond, and MMLU pro. For MMLU, the gap shrinks to just 0.5 percentage points (82.9% vs. 83.4%). For MATH-500, HumanEval, LiveCodeBench, the accuracy of Yuan3.0 Flash is close to DeepSeek-V3-0324. These findings establish Yuan3.0 Flash as a competitive alternative to larger models in core reasoning tasks, delivering comparable accuracy with a fraction of the parameters.

In the thinking mode, we analyze the performance of Yuan3.0 Flash compared with DeepSeek-R1-0528 across a wide range of benchmarks as shown in Table 10. For MATH-500, HumanEval, MMLU, the accuracy of Yuan3.0 Flash is close to DeepSeek-R1-0528, while consuming up to nearly 1/4 tokens. For AIME2024, GPQA-Diamond, and MMLU pro, the accuracy lags behind DeepSeek-R1-0528, while the tokens of Yuan3.0 Flash is up to around 1/3 tokens. In conclusion, the results establish that Yuan3.0 Flash is robustly capable of complex reasoning with superior token efficiency.

Table 8: Comparison among Yuan3.0 Flash and other models on visual benchmarks

(a) Comparison of performance in Non-Thinking settings			
	Yuan3.0 Flash Non-Thinking Mode	Qwen3-VL-32B- Instruct	Qwen2.5-VL- 72B
ChartQA	87.9	88.5	89.5
DocVQA	90.1	96.9	96.4
AI2D	85.7	89.5	88.7
MathVista	72.8	83.8	74.8

(b) Comparison of performance in Thinking settings			
	Yuan3.0 Flash Thinking Mode	Qwen3-VL-32B- Thinking	Qwen3-VL-235B- A22B-Thinking
ChartQA	90.1	89.0	90.3
DocVQA	90.2	96.1	96.5
AI2D	86.9	88.9	89.2
MathVista	74.1	85.9	85.8

(c) Comparison of average number of tokens in Thinking settings			
	Yuan3.0 Flash Thinking Mode	Qwen3-VL-32B- Thinking	Qwen3-VL-235B- A22B-Thinking
ChartQA	341	602	802
DocVQA	180	217	375
AI2D	427	1,234	1,387
MathVista	885	1,761	1,665

Table 9: Accuracy comparison of Yuan3.0 Flash and DeepSeek-V3-0324 under non-thinking mode

Benchmarks	DeepSeek-V3-0324 671B	Yuan3.0 Flash 40B
AIME 2024	59.4	32.6
MATH-500	94.0	88.7
LiveCodeBench	35.3	22.5
HumanEval	92.9	86.8
GPQA-Diamond	68.4	37.4
MMLU	83.4	82.9
MMLU pro	81.2	64.2

Table 10: Accuracy comparison of Yuan3.0 Flash and DeepSeek-R1-0528 under thinking mode.

Model	AIME 2024		MATH-500		LiveCodeBench		HumanEval		GPQA-Diamond		MMLU		MMLU pro	
	Acc	Tokens	Acc	Tokens	Acc	Tokens	Acc	Tokens	Acc	Tokens	Acc	Tokens	Acc	Tokens
DeepSeek-R1-0528	91.4	17,164	97.4	5,541	73.3	17,896	96.9	3,135	81.0	10,081	88.7	1,616	85.0	4,501
Yuan3.0 Flash 40B	47.6	6,086	91.2	1,431	29.8	8,157	95.6	2,313	47.4	3,761	83.7	806	68.1	1,688



## 7 Conclusion

In this paper, we introduce Yuan3.0 Flash, a 40B MoE multimodal language model with hybrid of thinking and non-thinking mode, is built to enhance performance on enterprise tasks while maintaining competitive capabilities on general purpose tasks. We propose RAPO algorithm that effectively mitigates the overthinking issue of LRMs, significantly improves training efficiency and model accuracy. Yuan3.0 Flash demonstrates superior performance in enterprise scenarios such as RAG, complex table processing, etc. Moreover, it also demonstrates strong reasoning capabilities in mathematics, science, etc., attaining accuracy comparable to frontier models while requiring only approximately 1/4 to 1/2 of the average tokens.

## References

- [1] Shukang Yin, Chaoyou Fu, Sirui Zhao, Ke Li, Xing Sun, Tong Xu, and Enhong Chen. A survey on multimodal large language models. *National Science Review*, 11(12):nwae403, 2024.
- [2] Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.
- [3] Jiayang Wu, Wensheng Gan, Zefeng Chen, Shicheng Wan, and Philip S Yu. Multimodal large language models: A survey. In *2023 IEEE International Conference on Big Data (BigData)*, pages 2247–2256. IEEE, 2023.
- [4] Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.
- [5] Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibao Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, et al. Qwen2. 5-vl technical report. *arXiv preprint arXiv:2502.13923*, 2025.
- [6] Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.
- [7] Zhe Chen, Jiannan Wu, Wenhai Wang, Weijie Su, Guo Chen, Sen Xing, Muyan Zhong, Qinglong Zhang, Xizhou Zhu, Lewei Lu, et al. Internvl: Scaling up vision foundation models and aligning for generic visual-linguistic tasks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 24185–24198, 2024.
- [8] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.
- [9] Qiguang Chen, Libo Qin, Jinhao Liu, Dengyun Peng, Jiannan Guan, Peng Wang, Mengkang Hu, Yuhang Zhou, Te Gao, and Wanxiang Che. Towards reasoning era: A survey of long chain-of-thought for reasoning large language models. *arXiv preprint arXiv:2503.09567*, 2025.
- [10] Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.
- [11] Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 26296–26306, 2024.
- [12] Zhuosheng Zhang, Aston Zhang, Mu Li, Hai Zhao, George Karypis, and Alex Smola. Multimodal chain-of-thought reasoning in language models. *arXiv preprint arXiv:2302.00923*, 2023.
- [13] Chenwei Lou, Zewei Sun, Xinnian Liang, Meng Qu, Wei Shen, Wenqi Wang, Yuntao Li, Qingping Yang, and Shuangzhi Wu. Adacot: Pareto-optimal adaptive chain-of-thought triggering via reinforcement learning. *arXiv preprint arXiv:2505.11896*, 2025.
- [14] Yongjiang Liu, Haoxi Li, Xiaosong Ma, Jie Zhang, and Song Guo. Think how to think: Mitigating overthinking with autonomous difficulty cognition in large reasoning models. *arXiv preprint arXiv:2507.02663*, 2025.
- [15] Shaohua Wu, Xudong Zhao, Shenling Wang, Jiangang Luo, Lingjun Li, Xi Chen, Bing Zhao, Wei Wang, Tong Yu, Rongguo Zhang, et al. Yuan 2.0: A large language model with localized filtering-based attention. *arXiv preprint arXiv:2311.15786*, 2023.
- [16] Qiyang Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.

- [17] Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, et al. Kimi k1. 5: Scaling reinforcement learning with llms. *arXiv preprint arXiv:2501.12599*, 2025.
- [18] Shenzhi Wang, Le Yu, Chang Gao, Chujie Zheng, Shixuan Liu, Rui Lu, Kai Dang, Xionghui Chen, Jianxin Yang, Zhenru Zhang, et al. Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for llm reasoning. *arXiv preprint arXiv:2506.01939*, 2025.
- [19] Deheng Ye, Zhao Liu, Mingfei Sun, Bei Shi, Peilin Zhao, Hao Wu, Hongsheng Yu, Shaojie Yang, Xipeng Wu, Qingwei Guo, et al. Mastering complex control in moba games with deep reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 6672–6679, 2020.
- [20] Zijun Liu, Peiyi Wang, Runxin Xu, Shirong Ma, Chong Ruan, Peng Li, Yang Liu, and Yu Wu. Inference-time scaling for generalist reward modeling. *arXiv preprint arXiv:2504.02495*, 2025.
- [21] Ganqu Cui, Lifan Yuan, Ning Ding, Guanming Yao, Bingxiang He, Wei Zhu, Yuan Ni, Guotong Xie, Ruobing Xie, Yankai Lin, et al. Ultrafeedback: Boosting language models with scaled ai feedback. *arXiv preprint arXiv:2310.01377*, 2023.
- [22] Junsoo Park, Seungyeon Jwa, Ren Meiyang, Daeyoung Kim, and Sanghyuk Choi. Offsetbias: Leveraging debiased data for tuning evaluators. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 1043–1067, 2024.
- [23] Chris Yuhao Liu, Liang Zeng, Jiakai Liu, Rui Yan, Jujie He, Chaojie Wang, Shuicheng Yan, Yang Liu, and Yahui Zhou. Skywork-reward: Bag of tricks for reward modeling in llms. *arXiv preprint arXiv:2410.18451*, 2024.
- [24] Zhilin Wang, Alexander Bukharin, Olivier Delalleau, Daniel Egert, Gerald Shen, Jiaqi Zeng, Oleksii Kuchaiev, and Yi Dong. Helpsteer2-preference: Complementing ratings with preferences. *arXiv preprint arXiv:2410.01257*, 2024.
- [25] Lianghai Zhu, Xinggang Wang, and Xinlong Wang. Judgelm: Fine-tuned large language models are scalable judges. *arXiv preprint arXiv:2310.17631*, 2023.
- [26] Tianyu Yu, Haoye Zhang, Qiming Li, Qixin Xu, Yuan Yao, Da Chen, Xiaoman Lu, Ganqu Cui, Yunkai Dang, Taiwan He, et al. Rlaif-v: Open-source ai feedback leads to super gpt-4v trustworthiness. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 19985–19995, 2025.
- [27] Tianyu Yu, Yuan Yao, Haoye Zhang, Taiwan He, Yifeng Han, Ganqu Cui, Jinyi Hu, Zhiyuan Liu, Hai-Tao Zheng, Maosong Sun, et al. Rlhfv: Towards trustworthy mllms via behavior alignment from fine-grained correctional human feedback. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13807–13816, 2024.
- [28] Yiyang Zhou, Chenhang Cui, Rafael Rafailov, Chelsea Finn, and Huaxiu Yao. Aligning modalities in vision large language models via preference fine-tuning. *arXiv preprint arXiv:2402.11411*, 2024.
- [29] Yi-Fan Zhang, Tao Yu, Haochen Tian, Chaoyou Fu, Peiyan Li, Jianshu Zeng, Wulin Xie, Yang Shi, Huanyu Zhang, Junkang Wu, et al. Mm-rlhf: The next step forward in multimodal llm alignment. *arXiv preprint arXiv:2502.10391*, 2025.
- [30] Lei Li, Zhihui Xie, Mukai Li, Shunian Chen, Peiyi Wang, Liang Chen, Yazheng Yang, Benyou Wang, Lingpeng Kong, and Qi Liu. Vlfeedback: A large-scale ai feedback dataset for large vision-language models alignment. *arXiv preprint arXiv:2410.09421*, 2024.
- [31] Yujie Lu, Dongfu Jiang, Wenhui Chen, William Yang Wang, Yejin Choi, and Bill Yuchen Lin. Wildvision: Evaluating vision-language models in the wild with human preferences. *Advances in Neural Information Processing Systems*, 37:48224–48255, 2024.
- [32] Jeffrey Li, Alex Fang, Georgios Smyrnis, Maor Ivgi, Matt Jordan, Samir Yitzhak Gadre, Hritik Bansal, Etash Guha, Sedrick Scott Keh, Kushal Arora, et al. Datacomp-1m: In search of the next generation of training sets for language models. *Advances in Neural Information Processing Systems*, 37:14200–14282, 2024.
- [33] Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*, 2021.
- [34] Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. Stanford alpaca: An instruction-following llama model, 2023.
- [35] Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. Making the v in vqa matter: Elevating the role of image understanding in visual question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6904–6913, 2017.

- [36] Drew A Hudson and Christopher D Manning. Gqa: A new dataset for real-world visual reasoning and compositional question answering. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6700–6709, 2019.
- [37] Amanpreet Singh, Vivek Natarajan, Meet Shah, Yu Jiang, Xinlei Chen, Dhruv Batra, Devi Parikh, and Marcus Rohrbach. Towards vqa models that can read. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8317–8326, 2019.
- [38] Anand Mishra, Shashank Shekhar, Ajeet Kumar Singh, and Anirban Chakraborty. Ocr-vqa: Visual question answering by reading text in images. In *2019 international conference on document analysis and recognition (ICDAR)*, pages 947–952. IEEE, 2019.
- [39] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.
- [40] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, et al. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International journal of computer vision*, 123(1):32–73, 2017.
- [41] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *Advances in neural information processing systems*, 36:34892–34916, 2023.
- [42] Guangxuan Xiao, Jiaming Tang, Jingwei Zuo, Junxian Guo, Shang Yang, Haotian Tang, Yao Fu, and Song Han. Duoattention: Efficient long-context llm inference with retrieval and streaming heads. *arXiv preprint arXiv:2410.10819*, 2024.
- [43] Hugo Laurençon, Andrés Marafioti, Victor Sanh, and Léo Tronchon. Building and better understanding vision-language models: insights and future directions. *arXiv preprint arXiv:2408.12637*, 2024.
- [44] Zihan Liu, Wei Ping, Rajarshi Roy, Peng Xu, Chankyu Lee, Mohammad Shoeybi, and Bryan Catanzaro. Chatqa: Surpassing gpt-4 on conversational qa and rag. *Advances in Neural Information Processing Systems*, 37:15416–15459, 2024.
- [45] Prasham Yatinkumar Titiya, Jainil Trivedi, Chitta Baral, and Vivek Gupta. Mmtbench: A unified benchmark for complex multimodal table reasoning. *arXiv preprint arXiv:2505.21771*, 2025.
- [46] Alexander R Fabbri, Wojciech Kryściński, Bryan McCann, Caiming Xiong, Richard Socher, and Dragomir Radev. Summeval: Re-evaluating summarization evaluation. *Transactions of the Association for Computational Linguistics*, 9:391–409, 2021.
- [47] Shishir G Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E Gonzalez. The berkeley function calling leaderboard (bfcl): From tool use to agentic evaluation of large language models. In *Forty-second International Conference on Machine Learning*, 2025.
- [48] Aniruddha Kembhavi, Mike Salvato, Eric Kolve, Minjoon Seo, Hannaneh Hajishirzi, and Ali Farhadi. A diagram is worth a dozen images. In *European conference on computer vision*, pages 235–251. Springer, 2016.
- [49] Ahmed Masry, Xuan Long Do, 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, 2022.
- [50] Minesh Mathew, Dimosthenis Karatzas, and CV Jawahar. Docvqa: A dataset for vqa on document images. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 2200–2209, 2021.
- [51] Pan Lu, Xiaoyi Ding, Lingfeng Wang, Zheng Zhu, and Junxian He. Mathvista: evaluating mathematical reasoning in visual contexts. *arXiv preprint arXiv:2309.09979*, 2023.
- [52] Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let’s verify step by step. In *The Twelfth International Conference on Learning Representations*, 2023.
- [53] MAA Codeforces. American invitational mathematics examination-aime 2024, 2024.
- [54] Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*, 2024.
- [55] Mark Chen. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.
- [56] Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyang Jiang, et al. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *Advances in Neural Information Processing Systems*, 37:95266–95290, 2024.

- [57] David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*, 2024.

## A Adaptive Image Segmentation for High-Resolution Visual Inputs

To mitigate geometric distortion and fine-grained detail loss when processing high-resolution images in MLLMs, we adopt an **Adaptive Image Segmentation Strategy** formulated as a dynamic programming optimization problem. Given an input image with resolution  $(W_l, H_l)$  and the fixed input resolution of the visual encoder  $(W_v, H_v)$ , the algorithm automatically determines the optimal number of image slices and their grid configuration. The objective is to maximally preserve the original geometric structure while maintaining computational efficiency.

To accommodate diverse image aspect ratios, we predefine a set of allowable slice counts,

$$N_{all} = \{1, 2, \dots, 9\} \quad (6)$$

For each candidate slice number  $N_i \in N_{all}$ , all feasible grid configurations  $(m, n)$  satisfying  $m \times n = N_i$  are enumerated. This yields a collection of candidate segmentation schemes:

$$C_N = \{m \times n = N_i, N_i \in N_{all}\} \quad (7)$$

where  $m$  denotes the number of columns (horizontal partitions) and  $n$  denotes the number of rows (vertical partitions).

To ensure that the segmented patches preserve the geometric proportions of the original image and to minimize distortions introduced by non-uniform resizing, we define a shape consistency score  $S(\cdot)$  that measures the discrepancy between the aspect ratio of the original image and that of the segmentation grid:

$$S(W_l, H_l, m, n) = \left| \frac{W_l}{H_l} - \frac{m}{n} \right| \quad (8)$$

A smaller  $S$  indicates a closer match between the grid layout and the original image aspect ratio, and thus lower geometric distortion.

However, optimizing shape consistency alone may result in excessive segmentation, particularly for small images, leading to unnecessary up-sampling and increased computational overhead. To address this issue, we introduce a regularization threshold  $\tau$  to constrain the expansion ratio. Specifically, we define the expansion ratio difference:

$$d(m, n) = \left| \frac{(W_v \times n + H_v \times m) - (W_l + H_l)}{W_l + H_l} \right| \quad (9)$$

Only candidate schemes satisfying  $d(m, n) < \tau$  are retained, forming a filtered candidate set  $C'_N$ .

If  $C'_N \neq \emptyset$ , the input image is considered too small to benefit from segmentation, and a default  $(1, 1)$  configuration is applied, i.e., no segmentation is performed. Otherwise, the optimal segmentation scheme  $(m^*, n^*)$  is selected by minimizing the shape consistency score:

$$(m^*, n^*) = \operatorname{argmin}_{(m, n) \in C'_N} S(W_l, H_l, m, n) \quad (10)$$

In cases where multiple candidate schemes yield identical  $S$  values, a tie-breaking rule is applied based on the effective resolution after segmentation. Specifically, if a scheme results in a total segmented area less than 50% of the original image area, i.e.,

$$0.5 \times (W_v \times H_v \times n \times m) < W_l \times H_l \quad (11)$$

it is preferred, as this avoids ineffective segmentation that primarily introduces redundant up-sampling.

Once the optimal segmentation parameters  $(m^*, n^*)$  are determined, the following preprocessing steps are performed:

1. **Local Slices:** The original image is partitioned into an  $(m^*, n^*)$  grid, and each local patch is resized to the visual encoder’s input resolution  $(W_v, H_v)$ .
2. **Global Thumbnail:** The original image is directly resized to  $(W_v, H_v)$  to provide a global contextual view.
3. **Feature Concatenation:** All local patches and the global thumbnail are concatenated to form the final visual input sequence for the multimodal model.

This adaptive image segmentation strategy enables efficient utilization of high-resolution visual information while preserving both global structure and local details, thereby improving visual grounding in MLLMs.

## B Adaptive Dynamic Sampling

---

**Algorithm 1** Adaptive Dynamic Sampling
 

---

**Input:**

Train batch  $\mathcal{B}$ ; Global batch size  $gbs$  (eg. 512); Mini batch size  $mb$ s (eg. 64)

**Output:**

resampled train batch  $\mathcal{B}_r$

Initialize prompts buffer  $Q$  for filtered prompts

Initialize pass rates buffer  $P_a$  for pass rates of  $Q$

**Step 1: Filter train batch  $\mathcal{B}$** 

**for**  $q_j$  **in**  $\mathcal{B}$  **do**

**if** all outputs of  $q_j$  are correct or incorrect **then**

        filter  $q_j$

**else**

$q_j \rightarrow Q$

$p_j = \frac{1}{G} \sum_{i=1}^G \mathbb{I}(a_i = 1)$

$p_j \rightarrow P_a$

▷ Compute pass rate of  $q_j$

**Step 2: Sort  $Q$  by pass rate**

Sort  $Q$  in descending order of pass rate

**Step 3: Compute the target size for completion**

$k = \lfloor \frac{|Q| + mb - 1}{mb} \rfloor$

▷ Compute the smallest multiples of  $mb$

$T = k \times mb$

▷ Compute target batch size

**Step 4: Complete train batch by resampling**

select the top  $N$  prompts  $Q^*$  in  $Q$

$combine(Q^*, Q) \rightarrow \mathcal{B}_r$

**return**  $\mathcal{B}_r$

---