
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>.

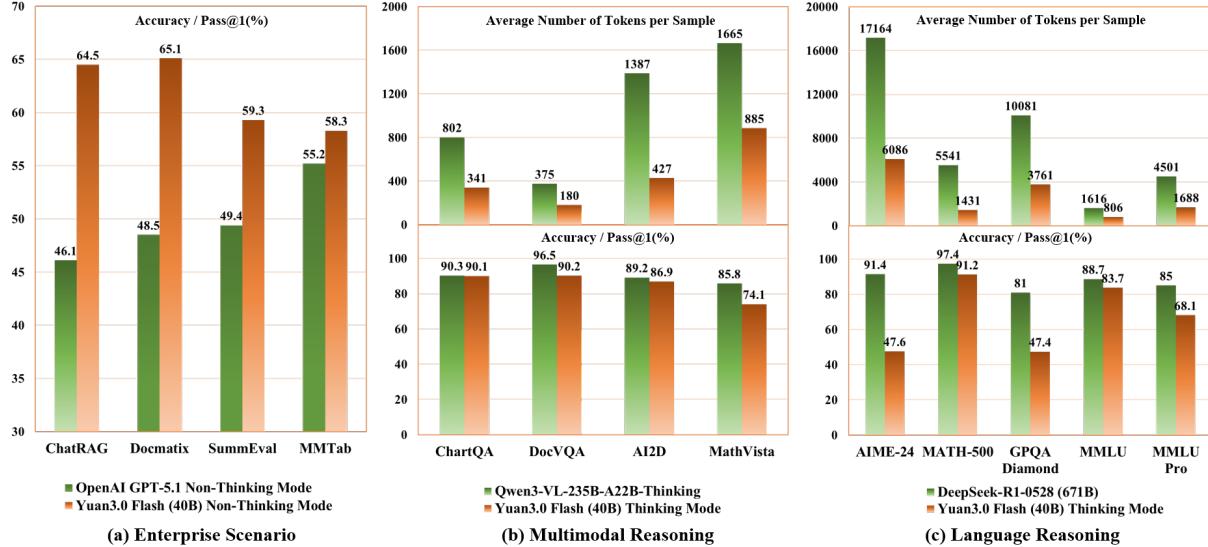


Figure 1: Performance–Efficiency Comparison of Yuan3.0 Flash across (a) Enterprise, (b) Multimodal, and (c) Language reasoning Benchmarks.

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 about large models, we carry out the following work:

- A high-quality data generation system is developed, 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.
- We propose a reflection-aware policy optimization (RAPO) reinforcement learning algorithm that effectively mitigates the overthinking issue of Large Reasoning Models (LRMs), improves training efficiency and model accuracy.
- We 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

Adopting the similar model architecture as the existing MLLM series models [7, 15, 16, 17], 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. This approach allows the parameter scale to grow approximately linearly while keeping training and inference FLOPs effectively controlled. The underlying architecture incorporates Localizing Filtering-based Attention [18], introducing an explicit inductive bias into self-attention by favoring local token dependencies that are prevalent in natural language. It augments the pairwise attention weight computation with two consecutive one-dimensional convolutions, enabling each token to model interactions primarily with a fixed number of preceding tokens while preventing future information leakage. Through this localized filtering mechanism, LFA effectively balances the global modeling capacity of self-attention with stronger sensitivity to short-range contextual structure.

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.

Pretained 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 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

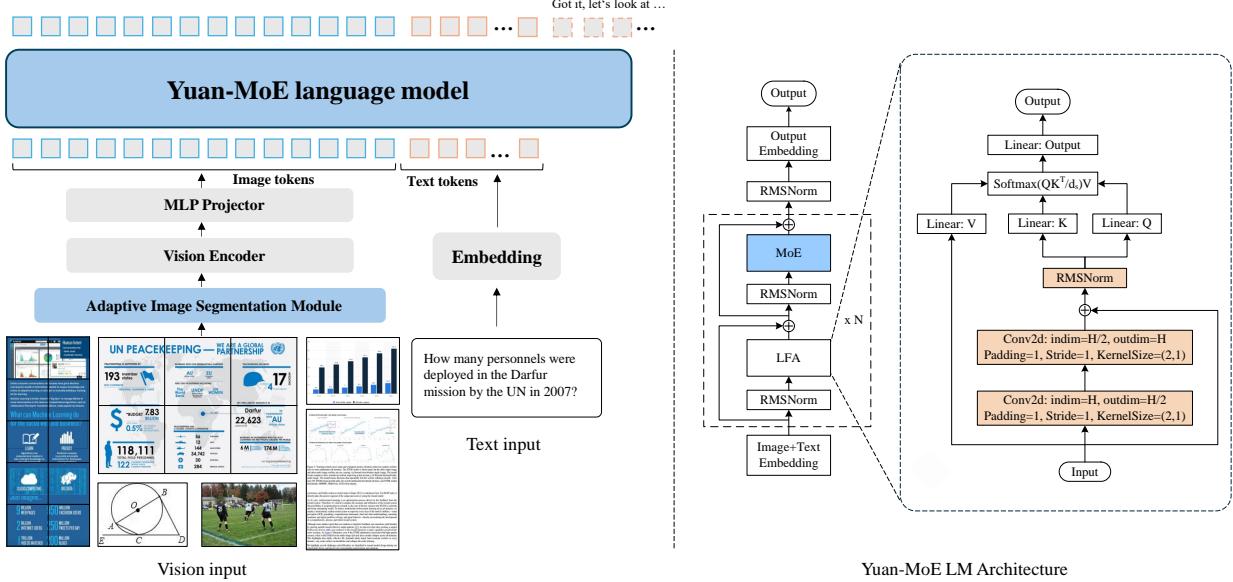


Figure 2: Overall architecture of Yuan3.0 Flash and MoE-based Language Backbone. The left part of the illustration depicts the proposed Yuan3.0 architecture, which comprises three integral components: (1) a ViT Encoder responsible for processing and encoding input images; (2) a lightweight MLP projector 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 part presents the language backbone with Localizing Filtering-based Attention (LFA)

Mechanism (RIRM), RAPO effectively mitigates the common "overthinking" problem in Large Reasoning Models (LRMs). Furthermore, we optimize the DAPO algorithm [19], developing adaptive training strategies and a unified reward system that supports multi-tas RL training with a hybrid of deep thinking and fast thinking. These innovations enable stable and efficient large-scale RL training, significantly enhancing Yuan3.0 Flash's performance across both objective evaluation metrics and subjective user experience dimensions.

3.1 Reflection Inhibition Reward Mechanism (RIRM)

LRMs driven by reinforcement learning with verifiable rewards (RLVR) generate extended reasoning chains to enhance inference capabilities. Models such as OpenAI's o1 [10], DeepSeek-R1 [8], and Kimi-1.5 [20] have demonstrated the effectiveness of this approach. However, the RLVR mechanism inadvertently encourages models to produce redundant reasoning steps to avoid errors, resulting in the "overthinking" phenomenon—an issue that not only wastes computational resources but may also degrade performance, becoming 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 we term "reflection". Our statistical analysis of Deepseek-R1 and DeepseekR1-Distill-1.5B [8] model on the AIME 2024 and MATH-500 benchmarks reveals that a substantial portion of token consumption occurs during this post-correct-answer reflection phase (Figure 3(b)).

To address "overthinking" problem, we propose the Reflection Inhibition Reward Mechanism (RIRM). RIRM first identifies the step where the model initially arrives at the correct answer and its subsequent reflection steps during the reasoning process. It then evaluates the reasoning trajectory based on three key dimensions: whether the correct answer appears before reflection, 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:

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 $s+2$ 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 $s + \frac{1}{2}$ km/h, including the same t minutes in the coffee shop.
But the question asks for the number of minutes. $0.3 \text{ hours} * 60 \text{ minutes/hour} = 3 * 60 = 180$, plus $0.4 * 60 = 24$, so 204 minutes.

What is it? But let me see the question: "the number of minutes the walk takes her, including the t minutes".
So 204 minutes. But let me confirm. $0.3 \text{ hours} * 60 = 180$. $0.4 * 60 = 24$. Total time 204 minutes.

The problem gave $s+2$, which is 0.5, and I got $s=2.5$, so $s+0.5=3$, which is. Is there 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. Total time 3 hours + 24 minutes = 180 + 24 = 204 minutes.

The problem says "the walk takes her", and includes the coffee time, so yes. 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.

From the expression $T = 4 + 9/(s+0.5) - 9/s$. $9/s = 9/2.5 = 3.6$. $9/(s+0.5) = 9/3 = 3$. So $T = 4 + 3 - 3.6 = 7.6 - 3.4 = 4.2$, same. With the quadratic, I got only one positive solution. In the discriminant, I had 784, sqrt 28, etc. $(s+2)^2 = 45/4 = 11.25$. $9/s = 11.25$. $s = 9/11.25 = 0.818$. But anyway, I think it's correct. So the answer should be 204 minutes.

</think>

Summary:

Aya walks a distance of 9 kilometers and stops at a coffee shop for $\frac{t}{60}$ minutes each time. Walking speed s km/h. Walking time = $\frac{9}{s}$ hours = 180 minutes. Coffee shop time = $\frac{t}{60}$ minutes. Total time = $\frac{9}{s} + \frac{t}{60} = 204$ minutes. The total time, including the coffee shop stop, is 204 minutes.

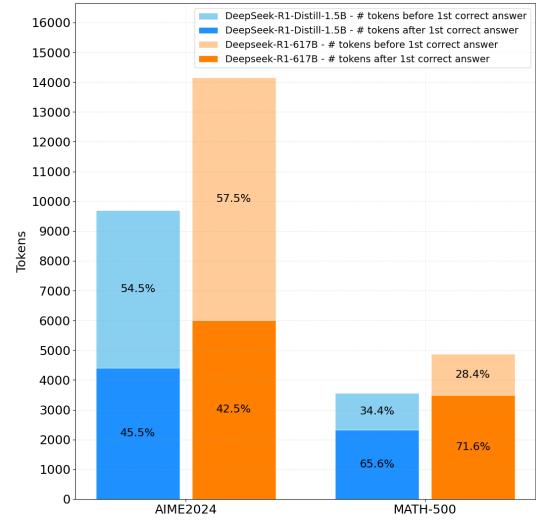
Correct answer

Reflection #1

Reflection #2

Reflection #3

Reflection #7



(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 AIMIE 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.

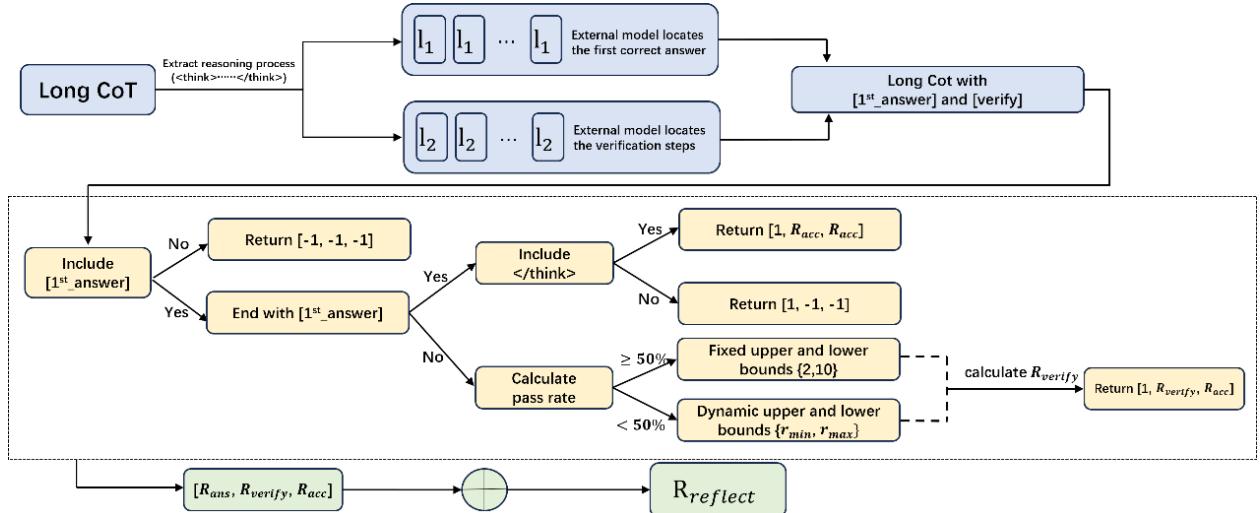


Figure 4: Illustration of Reflection Inhibition Reward Mechanism.

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 “[1st_answer]” and “[verify]” respectively.

2. Calculate three scores using the annotated reasoning trajectory:

- R_{ans} : whether the correct answer appears before reflection.
- R_{ver} : whether the number of reflections, v , is reasonable. For R_{ver} calculation, samples with $\geq 50\%$ rate use fixed bounds $r_{min} = 2$, $r_{max}=10$, while others use dynamic bounds $r_{min} = \tilde{r}_{min}$, $r_{max} = \tilde{r}_{max}$, with \tilde{r}_{min} and \tilde{r}_{max} are determined based on the minimum and maximum number of “[verify]” among N samples of the

same prompt. With the upper and lower bounds, R_{ver} is calculated in the following way:

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

- 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:

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

Incorporating RIRM into Yuan3.0 Flash training enables simultaneous improvements in model accuracy and output compression. Table 1 presents a comparison of accuracy and output length (in tokens) on AIME 2024 and MATH-500 between the Yuan3.0 40B SFT base model trained with DAPO length penalty and with RIRM. Figure 5 demonstrates the impact of RL training with RIRM, showing that the reasoning process of DeepseekR1-Distill-1.5B becomes significantly more concise. Notably, the token consumption during the reflection phase is substantially reduced.

Table 1: Performance comparison of the Yuan3.0 Flash (40B) model under RL training with explicit DAPO length penalty versus the proposed RIRM. Compared to the baseline model, the RIRM-based reinforcement learning framework yields a maximum accuracy improvement of 52.37% while simultaneously reducing output length by up to 47.14%.

Method	AIME 2024		MATH-500	
	Accuracy	Length	Accuracy	Length
Yuan3.0 Flash (40B) SFT	31.45	13656	83.20	3362
RL+DAPO length-penalty	46.35	13781	89.06	3974
RL+RIRM	47.92	7505	89.47	1777

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. $25^\circ\text{C} + 273.15 = 298.15 \text{ K}$
 $\approx 300 \text{ K}$.<|boxed{C}|<|end_of_sentence|><eod>

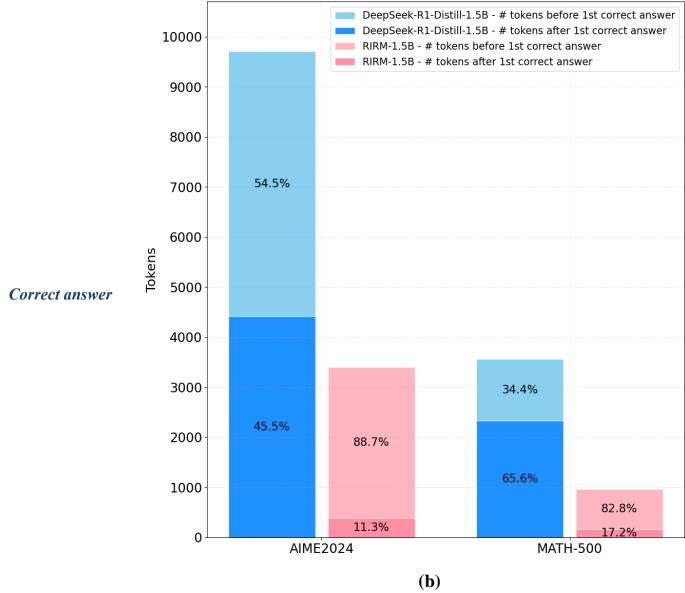


Figure 5: (a) Example of DeepseekR1-Distill-1.5B’s reasoning process after trained with RIRM. (b) Comparison of average token consumption of DeepseekR1-Distill-1.5B before and after trained with RIRM in AIME 2024 and MATH-500 benchmarks. The light-colored segments indicate token consumption before the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

Table 2: We train DeepSeek-R1-Distill-Qwen-1.5B using DAPO and the enhanced DAPO with Adaptive Dynamic Sampling (ADS) on 60K mathematical reinforcement learning samples. The results show that DAPO with ADS achieves a significant reduction in both average sampling time and iteration time per step compared to the baseline DAPO, leading to a 52.91% improvement in training efficiency.

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

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 objective:

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

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—leading to data waste, increased iteration time, and the need for additional training epochs.

To improve sampling efficiency, we develop an 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 pass rates: $P_j = \frac{1}{G} \sum_{i=1}^G I(a_i = 1)$. We then sort prompts by pass 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.

80/20 rule DAPO encounters training instability issues when applied to large-scale Mixture-of-Experts (MoE) models. Building upon DAPO, we introduce the 80/20 algorithm [21], 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 [22] 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 loss using the following formula:

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

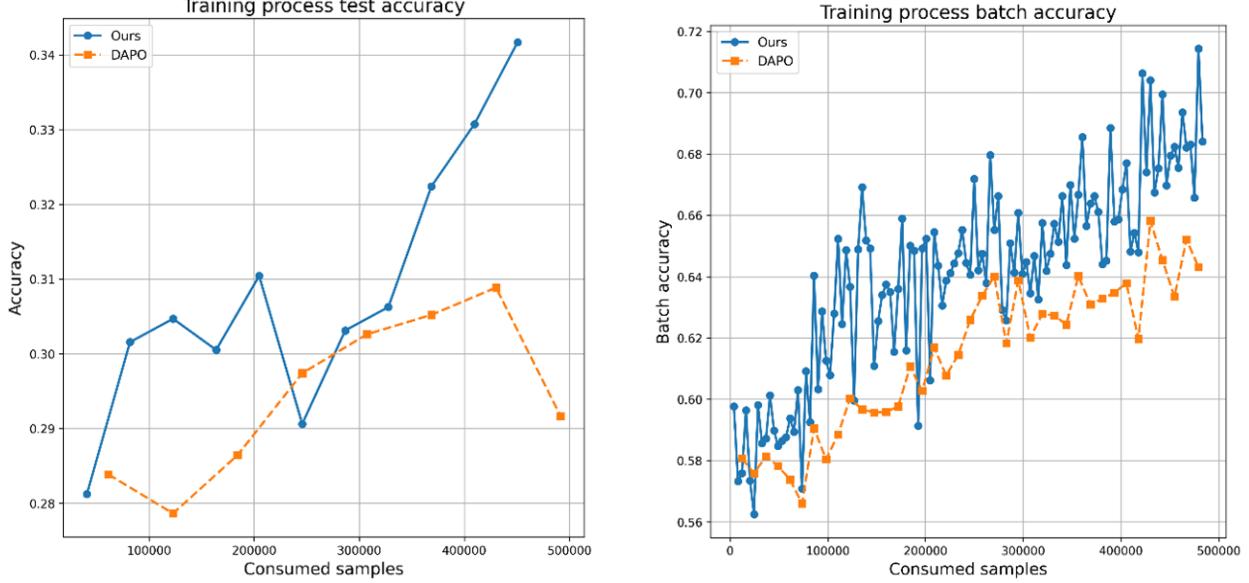


Figure 6: We train DeepSeek-R1-Distill-Qwen-1.5B with DAPO and DAPO equipped with ADS on 60K math RL samples. The curve illustrates the efficiency of the model’s accuracy improvement during training and testing.

where

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

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. τ_p is the corresponding entropy threshold, such that only tokens with $H_i^t \geq \tau_p$ are used for gradient computation; these tokens constitute the top ρ portion of all tokens in the response. B_{mini} denotes the mini-batch size, and after adaptive dynamic sampling, the global batch size B_r becomes k times the mini-batch size.

Shown in Figure 7, our optimized DAPO algorithm demonstrates greater stability during training, with moderate growth in length and a gradual increase in entropy. [19] posit that a gradual increase in entropy supports better model performance.

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 deep thinking tasks (including language and multimodal mathematical reasoning, code generation, and scientific tasks) and fast thinking tasks (such as language and multimodal RAG, language and multimodal chat, and general scenario QA). This unified approach maximizes the model’s generalization capability while maintaining task diversity.

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 deep thinking tasks and all fast 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

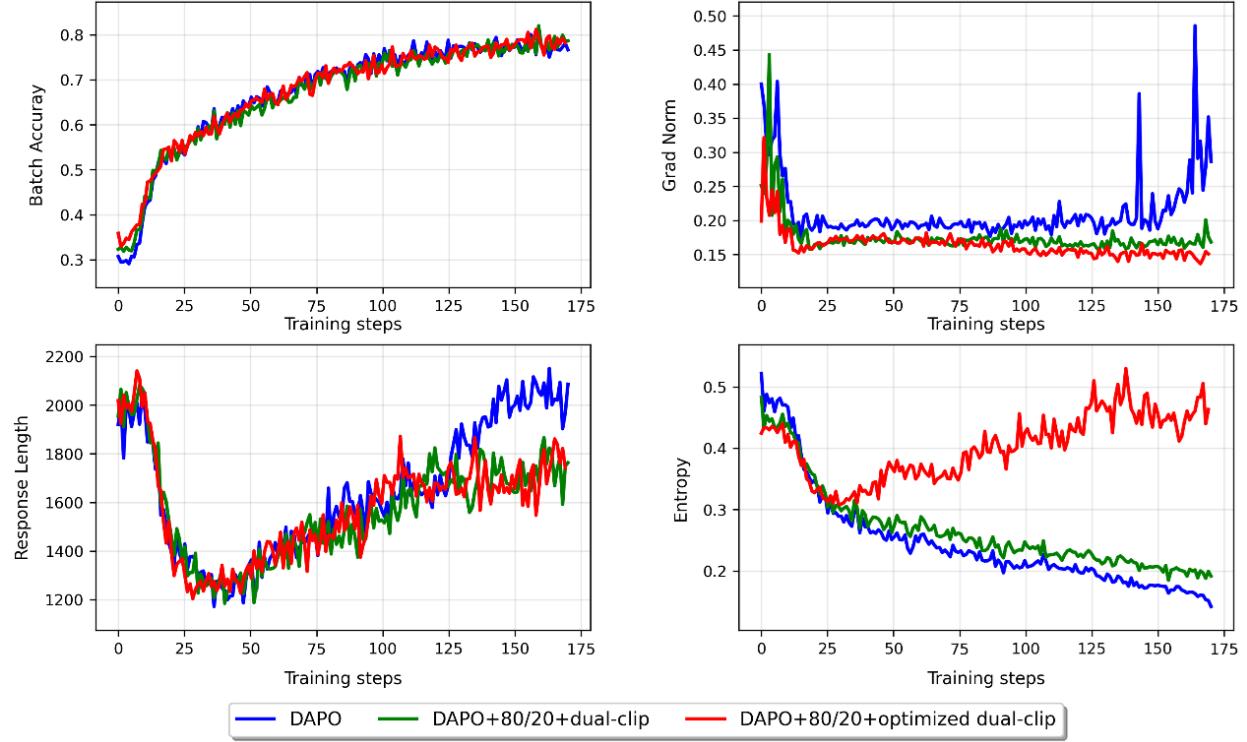


Figure 7: Training dynamics of DAPO (blue line), DAPO with 80/20 rule and dual-clip technique (green line), and DAPO with 80/20 rule and our optimized dual-clip technique.

longest one in a batch, severely hampering efficiency. To address this, we introduce a data grouping and alternating 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% (reducing time from 633 seconds to 529 seconds).

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 ("<|Assistantl>", "<lend_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 [23]. 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 (e.g., Qwen2.5-72B-Instruct) 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 [24], OffsetBias [25], Skywork-Reward [26], HelpSteer2 [27], JudgeLM [28]). 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 [29], RLHF-V [30], Povid [31], MM-RLHF [32], VL-Feedback [33], wildvision-battle [34]). A multimodal teacher model (e.g., QwenVL2.5-72B-Instruct) 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 pure text and image-text pairs was utilized to train the Yuan3.0 Base model, and the key focuses include the high quality of corpora as well as the enhancement of the model’s foundational capabilities for enterprise scenario.

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 - many of which exhibit higher linguistic rigor and more structured logical frameworks. Such characteristics help enhance the model’s adaptability and performance in specialized domains. To guide the curation of web-crawled data, we first trained a FastText-based domain classifier using a labeled domain

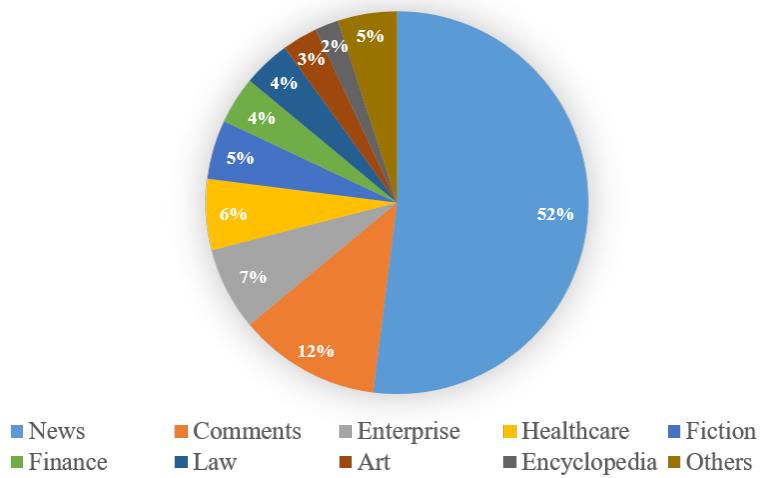


Figure 8: Domain-wise composition of the web-crawled pre-training corpus.

dataset. This classifier is capable of distinguishing across a broad spectrum of domains, including advertising, economy, healthcare, law, business, literature, culture, history, politics, art, entertainment, general knowledge, opinion, education, science, and technology. Through sampling analysis of open-source web-crawled datasets [35] by our domain classifier, it is indicated that content such as advertisements, lifestyle news, and entertainment/sports news accounts for the majority (Figure 8). If left unregulated, this would skew the data distribution, weaken the model’s professional competence in enterprise-oriented scenarios, and hinder its performance in domain-specific tasks.

We performed targeted selection on the web-crawled corpus. Firstly, we reduce the proportion of low-value news and advertising content while retaining high-value professional data. Next, we retained all web data from four high-value domains closely linked to enterprise demands: technology, science, opinion, and knowledge. Furthermore, we increased the proportion of industries such as finance, law, manufacturing, healthcare, information technology and education. We also trained 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.

The data covering sciences and various professional industries is specifically curated to strengthen the model’s cross-domain reasoning capabilities and professional competence. It encompasses content related to real-world mathematical applications as well as scientific data including experimental analysis, scientific deduction across physics and engineering. Additionally, industry-specific data tailored for enterprise scenarios is integrated, such as patent documentation, financial calculation, legal document parsing, educational knowledge reasoning. Furthermore, all code snippets covering multi-language programming in pre-training datasets are accompanied by comments in Chinese, English, or both. Among these, the codes covering enterprise-oriented scenarios is included, such as industrial control coding, financial system development, backend service framework construction, etc.

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 256 million 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.

Common Scenario dataset is collected from large-scale web-crawled image-text collections such as LAION-5B [36].

Code dataset from open-source dataset and focuses on programming comprehension and generation.

Document dataset collected from a widely used open-source corpus such as DocVQA [37].

OCR dataset is collected from open-source dataset such as OCR-VQA [38], providing a large volume of synthetic images with word-level annotations.

Mathematical Formula dataset for recognizing and translating rendered mathematical formulas into LaTeX format formulas.

Charts & Diagrams: For chart understanding, the dataset is collected from multiple open-source datasets such as PlotQA [39].

Autonomous Driving: The autonomous driving dataset is collected from multiple open-source datasets including nuScenes [40], which provides large-scale multimodal sensor data with detailed 3D annotations.

4.2 Fine-tuning datasets

In fine-tune stage, we introduce long chain of thought as a guide for generating answers for all language tasks that require "systematic thinking" (math, code, STEM etc.). For straightforward tasks such as chatting and knowledge QA, short chain of thought are employed. In order to guarantee the high quality of these data, we have established filtering and cleaning workflows for various task types, especially for the consideration of the accuracy, professionalism and security of enterprise-level downstream tasks.

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. Considering that in practical scenarios, the core requirement for the model is to professionally answer user queries accurately and reliably. Thus, the model conducts comprehensive evaluation focusing on four key dimensions: first, relevance, verifying whether the answer accurately matches the user's question without deviating from the topic or being irrelevant; second, logicality, assessing the coherence and organization of the answer to ensure clear reasoning and logical closure; third, professional normativity, ensuring that responses comply with the inherent constraints and expression standards of industries; fourth, security, strictly investigating vulgar, non-compliant, and misleading content to eliminate risky information. Finally, only samples with a comprehensive score reaching the preset threshold are retained, ensuring the quality and compliance of fine-tune data.

For fine-tuning data without existing verifiable standard answers temporarily, the standardized definition of standard answers is first completed through the method of "multi-model collaborative annotation & multi-sampling voting". Specifically, multiple basic models with different architectures are invoked for independent annotation of samples. For samples with significant discrepancies in annotation results, the annotation dimension is expanded through multiple sampling, and the optimal standard answer is determined by combining voting algorithms. After annotation, dual filtering is performed simultaneously: first, correctness verification is carried out based on the annotated standard answers; second, format unification is implemented (e.g., unified typesetting, field alignment, and redundant information removal) to generate high-quality data that meets fine-tuning requirements.

For multi-modal tasks related to understanding and interaction, a chain-of-thought-augmented VQA dataset spanning multiple domains (e.g. common scenario, documents, charts, visual reasoning problems, code, OCR, science problems) is constructed. This dataset was built upon publicly available VQA data, retaining the original images and questions while employing the LLM to generate step-by-step rationales. These generated reasoning chains were then integrated with the ground-truth answers to form enriched training examples that encapsulate complete reasoning processes.

4.2.2 Datasets for Enterprise Scenario

In the fine-tuning training, we further supplemented a wealth of training data highly relevant to enterprise application scenarios, including RAG, tabular data, text summarization, instruction following, Text2SQL, and tool invocation capabilities.

We construct a hybrid dataset integrating human annotations, synthetic generation, and open-source resources. It addresses the scarcity of high-quality supervision, modality imbalance, and the quality-scale trade-off, while ensuring both accuracy and broad coverage.

Human-annotated Data. We create 45K dialogues (25K text-only, 20K multimodal) via expert role-playing. These dialogues feature multi-turn contextual progression, enabling gradual context accumulation. Multimodal samples are aligned with visual inputs (e.g., code snippets, diagrams, API screenshots), enabling fine-grained grounding. The dataset serves as the foundation for supervised fine-tuning.

Synthetic Data. To address annotation bottlenecks, we generate synthetic data across 12 technical domains (e.g., Literature, Geography, Biology, Economics, Code) leveraging data sources such as Wikipedia, arXiv and Papers With

Code. The synthetic data includes 305K samples (160K text-only, 145K multimodal), which ensures semantic diversity, covers 95% of low-frequency concepts.

Open-source Data. We also include general instruction datasets [41, 42] and multimodal understanding datasets [43, 44, 45, 38, 46, 47, 17] covering 200K text samples and 160K multimodal samples, enhancing domain coverage and adding diverse real-world data.

4.3 Datasets for Reinforcement Learning

4.3.1 Datasets for General tasks

For language-related tasks, we have designed a standardized workflow to generate reinforcement learning datasets for corresponding domains. Firstly, data with clear standard answers were extracted as candidate seed data. For mathematical and scientific data, the standard answer is defined as the final, most simplified answer. For code-related data, the standard answer comprises several unit test cases (with more than 5 unit tests per problem). Secondly, for mathematical and other similar tasks, rule-based filtering was applied to remove items such as multiple-choice questions, multi-part questions, and proof-based questions. Next, the remaining seed data were graded by difficulty using the sft model (the model after fine-tuning) employed for reinforcement learning training. Specifically, five answers are sampled for each question, and only those questions with a pass rate of 1/5, 2/5, or 3/5 are retained for subsequent reinforcement learning training.

Furthermore, regarding multimodal tasks, we construct 70k high-quality samples by similar filtering process, strategically distributed across six distinct categories: Mathematics Problems, Common Scenarios, Charts & Diagrams, Scientific QA, Logic Puzzle and Document. All datasets are designed to advance the model's abstract visual cognition and relational reasoning ability.

4.3.2 Datasets for Enterprise Scenario

Human-annotated Data. We construct 18K data (10K text-only, 8K multimodal), annotated by domain specialists. Evaluations capture nuanced distinctions in technical accuracy, reasoning depth, contextual coherence, and visual-textual alignment, offering high-quality supervision for RL.

Synthetic Data. We scale data via an automated pipeline that generates synthetic data. After that, we leverage the post-SFT model to perform 5 rounds of inference on the constructed data to evaluate the difficulty level of the samples. For samples where correctness can be objectively judged (e.g., multiple-choice questions, numerical calculation problems), we select those answered correctly 1, 2, or 3 times as the final data. For text generation tasks (e.g., question answering, text summarization), we adopt metrics including BLEU, F1 score, and model-based scores to evaluate the correctness of results, and then sample different quantities of data from distinct score intervals. We ultimately obtain 152K high-quality samples (70K text-only, 82K multimodal).

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 enhancing the model's overall capabilities in tasks involving dialogue, comprehension, localization, and reasoning.

The third stage involves supervised fine-tuning based on high-quality instruction data, further improving the model's multimodal instruction-following and hierarchical reasoning capabilities.

Finally, large-scale reinforcement learning is conducted, continuously optimizing the model's reasoning consistency and behavior quality in real interaction scenarios via multimodal reward models and stabilization techniques. The RL training adopted a hybrid training mode integrating long and short thinking capabilities.

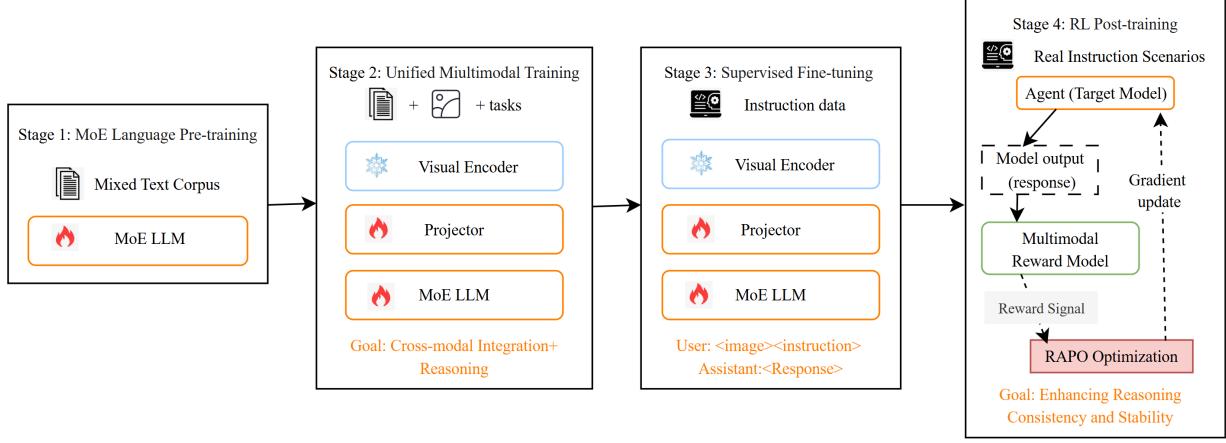


Figure 9: Structure of multi-stage collaborative multimodal training framework.



Figure 10: Pressure Testing Yuan3.0 Flash via “Needle in A HayStack”.

6 Experiments

Yuan3.0 is equipped with support for a 128K context length. To assess the model’s ability to accurately locate critical information within long contexts, we evaluate Yuan 3.0 Flash model using the Needle-in-a-Haystack (NIAH) benchmark [48] . The experiment was 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).

6.1 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 DocMatrix benchmark [49] 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 an accuracy of 65.1, substantially outperforming Claude 3.5 (42.6), GPT-4o (56.8), and GPT-o3 (45.6), and GPT-5.1 (48.5) (Table 3). These results demonstrate strong multimodal reasoning and retrieval capabilities in document-level scenarios.

For text retrieval, we evaluate on ChatRAG [50], a widely recognized industrial benchmark comprising ten tasks. These tasks include long-context retrieval (D2D, QuAC, QReCC), short-context and structured retrieval (CoQA, DoQA, CFQA, SQA, HDial), as well as Wikipedia-based retrieval tasks (TCQA, INSCIT). Across the full ChatRAG task set, Yuan3.0 Flash attains an average accuracy of 64.5, surpassing DeepSeek-V3, DeepSeek-R1, GPT-4o, and GPT-o3, and

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

Model	Acc(%)
Qwen2.5-VL-72B-Instruct	59.8
InternVL3-78B	43.0
Claude3.5-Sonnet	42.6
OpenAI GPT-4o	56.8
OpenAI GPT-o3	45.6
OpenAI GPT-5.1	48.5
Yuan3.0 Flash	65.1

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	50.5	31.6	28.9	49.3	77.0	26.1	83.5	82.1	46.7	47.4	32.1
OpenAI GPT-4o	50.5	32.8	26.6	49.3	76.1	28.8	81.9	81.1	49.8	41.3	26.7
OpenAI GPT-o3	44.1	23.1	20.8	40.4	69.4	18.6	67.8	86.7	45.9	41.3	26.7
DeepSeek-R1	39.4	21.5	22.2	42.4	62.5	24.7	81.5	82.1	30.7	38.0	28.7
OpenAI GPT-5.1	46.1	28.2	23.2	45.4	68.8	20.9	73.1	81.3	44.7	45.4	30.0
Yuan3.0 Flash	64.5	49.8	53.8	57.1	90.9	60.0	74.4	87.5	66.3	68.4	36.4

achieves leading performance on 8 out of the 10 tasks, indicating robust retrieval effectiveness across diverse textual contexts (Table 4).

Multimodal table understanding is a critical capability in enterprise office scenarios. We evaluate this dimension using MMTab [51], which consists of 15 authoritative benchmarks spanning multiple task categories. Question-answering tasks include TABMWP (table-based math word problems), WTQ and HiTab (complex querying and reasoning over Wikipedia tables), TAT-QA (joint reasoning over financial tables and text), and FeTaQA (free-form answer generation from tables). Fact-verification and text-generation tasks include TabFact, InfoTabs, HiTab_T2T, Rotowire, and WikiBIO, while TSD, TCE, TCL, MCD, and RCE focus on long-context table-centric information processing. Yuan3.0 Flash achieves a leading average accuracy of 58.3, 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).

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
Avg.		52.0	29.9	55.2	58.3
QA	TABMWP (Acc.%)	88.2	60.5	64.9	95.1
	WTQ (Acc.%)	77.4	48.0	60.8	68.2
	HiTab (Acc.%)	51.5	27.5	77.8	69.8
	TAT QA (Acc.%)	62.7	32.5	61.4	69.2
	FeTaQA (BLEU)	5.3	11.0	8.7	28.4
Fact Checking	TabFact (Acc.%)	89.4	45.5	52.8	87.3
	InfoTabs (Acc.%)	79.5	65.5	64.3	83.5
	HiTab T2T (BLEU)	5.2	3.0	44.2	13.3
	Rotowire (BLEU)	4.5	4.2	17.8	14.7
	WikiBIO (BLEU)	2.7	1.9	11.9	17.3
TSD	Row (Acc.%)	47.4	19.0	96.6	46.6
	Col (Acc.%)	89.7	38.0	62.1	82.8
TCE (Acc.%)		52.7	14.4	86.4	56.8
TCL (Acc.%)		50.8	27.9	44.7	57.0
MCD (F1%)		43.5	3.5	72.5	65.2
RCE	Row (F1%)	50.8	48.5	53.6	62.1
	Col (Acc.%)	82.8	57.1	57.2	73.7

For text summarization, which is essential for compressing user history in agent-based applications, we evaluate on SummEval [52], covering lexical overlap, semantic similarity, and factual consistency. Yuan3.0 Flash achieves an

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	59.3	25.5	9.2	86.3	68.2
DeepSeek-V3.2	51.4	33.3	11.9	85.6	41.8
Gemini-2.0-Flash	45.4	24.8	8.7	85.7	29.5
Claude-3.5-Sonnet	45.4	24.1	8.3	85.2	30.7
OpenAI GPT-4o	46.5	25.0	8.9	85.9	32.5
OpenAI GPT-5.1	49.4	27.5	10.2	84.6	40.5
Yuan3.0 Flash	59.3	51.3	28.3	90.0	45.3

average score of 59.3 (Table 6, outperforming Gemini (45.4) and GPT-5.1 (49.4), highlighting its strength in producing concise, semantically faithful, and factually consistent summaries.

Finally, tool invocation ability is assessed using BFCL V3 [53], which evaluates real-world function-calling competence across five tasks: Non-Live AST (static function selection and argument extraction), Live AST (dynamic execution with real-time feedback), Multi-turn (context maintenance and multi-tool coordination), Relevance Detection (deciding whether tool invocation is required), and Irrelevance Detection (rejecting invalid or unnecessary calls). Yuan3.0 Flash demonstrates consistently strong performance across all categories (58.0 on average), with no evident weaknesses, underscoring its reliability and maturity in tool-augmented reasoning and execution.

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	67.9	87.9	77.0	40.1	83.3	76.3
Claude-3.7-Sonnet	58.6	41.3	78.4	48.4	72.2	81.4
OpenAI GPT-4.1-nano	53.0	76.7	64.3	19.9	94.4	59.4
OpenAI o3-mini	51.3	42.1	77.3	26.1	77.8	80.7
Llama-4-Scout-17B-16E-Instruct	45.4	83.5	58.0	1.9	100.0	39.7
Yuan3.0 Flash	58.0	69.1	66.4	36.9	94.4	77.1

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.

6.2 Multimodal Benchmarks

To comprehensively evaluate the capabilities of Yuan3.0 Flash model across diverse visual-language understanding scenarios, we evaluate on the following benchmarks, categorized by their primary focus:

- AI2D [54] is a benchmark for reasoning over science diagrams, requiring models to parse visual elements and their relationships to answer questions.
- ChartQA [55] 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 [37] tests the understanding of scanned documents, requiring models to answer questions by interpreting textual content, layout, and visual cues within document images.
- MathVista [56] 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, Yuan3.0 Flash scored slightly behind other models but has comparable scores, reflecting a notable strength 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

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	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	1234	1387
MathVista	885	1761	1665

Flash Thinking Mode showed gains in some tasks compared to its non-thinking mode: 90.2 (+2.5) on ChartQA, 86.6 (+0.8) on AI2D, and 74.8 (+2.3) on MathVista, and DocVQA remained unchanged at 90.1 as shown in Table 8b. 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. For AI2D task, Yuan3.0 Flash achieves a slight increase from its 85.8 score in the non-thinking setting 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 overhead compared to the two Qwen3 models. The number of output tokens is substantially lower in all tasks, with the token volume saved reaching 1.2 to over 3 times in different benchmarks, effectively mitigating 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 cutting down token-based computational costs, which fully demonstrates its high reasoning efficiency.

6.3 Evaluation on General Reasoning Benchmarks

To evaluate mathematical reasoning and coding ability, we assessed performance on the following benchmarks: Math-500 [57], AIME 2024 [58], LiveCodeBench [59], and HumanEval [60]. AIME exam comprises two sections of 15 questions each. For every question, we generated 64 samples and reported the average accuracy as the final score. For Math-500, LiveCodeBench, and HumanEval, we generate 8 samples per question, with the final score computed as the average accuracy. We further evaluated general knowledge and advanced reasoning on MMLU, MMLU-Pro [61] and GPQA-Diamond [62]. For GPQA-Diamond, 8 samples were generated per question, and the average accuracy was used as the evaluation metric. In all the above evaluations, we set the temperature to 0.6, top-p to 0.95, and the maximum output token limit to 16,384 for thinking mode and 4,096 for non-thinking mode. The table below presents a

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
ATME 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

comparison of the Yuan3.0 Flash model with Deepseek-V3-0324 and Deepseek-R1-0528 under both non-thinking and thinking modes.

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

Benchmarks	Metrics	DeepSeek-R1-0528 671B	Yuan3.0 Flash 40B
AIME 2024	ACC	91.4	47.6
	MTPQ	17164	6086
	MTPA	187	128
MATH-500	ACC	97.4	91.2
	MTPQ	5541	1431
	MTPA	57	16
LiveCodeBench	ACC	73.3	29.8
	MTPQ	17896	8157
	MTPA	244	274
HumanEval	ACC	96.9	95.6
	MTPQ	3135	2313
	MTPA	32	24
GPQA-Diamond	ACC	81.0	47.4
	MTPQ	10081	3761
	MTPA	124	79
MMLU	ACC	88.7	83.7
	MTPQ	1616	806
	MTPA	18	10
MMLU pro	ACC	85	68.1
	MTPQ	4501	1688
	MTPA	53	25

In the non-thinking mode, the accuracy comparison across multiple benchmarks (Table 9) reveals that Yuan3.0 Flash (40B) demonstrates competitive performance relative to DeepSeek-V3-0324 (671B), with only marginal gaps observed in core reasoning and knowledge benchmarks. Specifically, on mathematical and coding tasks (Math-500, HumanEval), Yuan3.0 Flash trails DeepSeek-V3-0324 by several percentage points. For general knowledge (MMLU), the gap shrinks to just 0.5 percentage points (82.9% vs. 83.4%). These results indicate that Yuan3.0 Flash, despite its significantly smaller parameter scale, maintains strong alignment with the performance of a much larger model in fundamental reasoning and application scenarios.

In the thinking mode (Table 10), we analyze the performance of Yuan3.0 Flash against DeepSeek-R1-0528 across benchmarks, with metrics defined as: ACC (accuracy), MTPQ (mean tokens per question), and MTPA (mean tokens per question for unit accuracy). Yuan3.0 Flash demonstrates standout efficiency-performance balance in thinking-mode evaluations (Table 10). For example, for MATH-500, it achieves an accuracy of 91.2 (only 6.2 points below DeepSeek-R1-0528) while consuming just 1431 average tokens—a sharp reduction from DeepSeek-R1-0528’s 5541 tokens. This shows Yuan3.0 Flash can handle complex mathematical reasoning with far fewer computational resources, making it more cost-effective for inference-sensitive deployment scenarios.

Overall, Yuan3.0 Flash demonstrates balanced performance across mathematical reasoning, code understanding, and general knowledge tasks, with standout strengths in resource-efficient and high-precision reasoning, which are particularly valuable for practical deployment requiring both performance and cost control.

7 Conclusion

In this paper, we introduce Yuan3.0 Flash, a 40B MoE multimodal language model, hybrid of reasoning and non-reasoning mode, built for application in industries and enterprise scenario. We propose RAPO algorithm that effectively mitigates the overthinking issue of Large Reasoning Models (LRMs), improves training efficiency by up to 52.91%, and achieves enhancement in model accuracy by up to 52.37%. Yuan3.0 Flash demonstrates outstanding performance in key industry and enterprise application scenarios such as RAG, complex table processing, and summarization. Yuan3.0 Flash not only exhibits strong capabilities in both text-modal and multi-modal reasoning tasks but also reduces the number of inference tokens by up to 75%.

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):nwaef03, 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, Sibo 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 Schaeckermann, 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, Huawei 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] Ruizhe Chen Shuai Bai, Yuxuan Cai et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.
- [16] Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, et al. Qwen2-vl: Enhancing vision-language model’s perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*, 2024.

- [17] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *Advances in neural information processing systems*, 36:34892–34916, 2023.
- [18] 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.
- [19] Qiying 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.
- [20] 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.
- [21] 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.
- [22] 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.
- [23] 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.
- [24] 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.
- [25] Junsoo Park, Seungyeon Jwa, Ren Meiying, 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.
- [26] Chris Yuhao Liu, Liang Zeng, Jiacai 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.
- [27] 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.
- [28] Lianghui Zhu, Xinggang Wang, and Xinlong Wang. Judgelm: Fine-tuned large language models are scalable judges. *arXiv preprint arXiv:2310.17631*, 2023.
- [29] Tianyu Yu, Haoye Zhang, Qiming Li, Qixin Xu, Yuan Yao, Da Chen, Xiaoman Lu, Ganqu Cui, Yunkai Dang, Taiwen 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.
- [30] Tianyu Yu, Yuan Yao, Haoye Zhang, Taiwen He, Yifeng Han, Ganqu Cui, Jinyi Hu, Zhiyuan Liu, Hai-Tao Zheng, Maosong Sun, et al. Rlhf-v: 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.
- [31] 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.
- [32] 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.
- [33] 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.
- [34] 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.
- [35] 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-lm: In search of the next generation of training sets for language models. *Advances in Neural Information Processing Systems*, 37:14200–14282, 2024.

- [36] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, et al. Laion-5b: An open large-scale dataset for training next generation image-text models. *Advances in neural information processing systems*, 35:25278–25294, 2022.
- [37] 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.
- [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] Nitesh Methani, Pritha Ganguly, Mitesh M Khapra, and Pratyush Kumar. Plotqa: Reasoning over scientific plots. In *Proceedings of the ieee/cvf winter conference on applications of computer vision*, pages 1527–1536, 2020.
- [40] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11621–11631, 2020.
- [41] 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.
- [42] Rohan Taori, Ishaaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. Stanford alpaca: An instruction-following llama model, 2023.
- [43] 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.
- [44] 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.
- [45] 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.
- [46] 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.
- [47] 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.
- [48] 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.
- [49] 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.
- [50] 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.
- [51] 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.
- [52] 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.
- [53] 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.
- [54] 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.

- [55] 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.
- [56] Pan Lu, Xiaoyi Ding, Lingfeng Wang, Zheng Zhu, and Junxian He. Mathvista: evaluating mathematical reasoning in visual contexts. *arXiv preprint arXiv:2309.09979*, 2023.
- [57] 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.
- [58] MAA Codeforces. American invitational mathematics examination-aime 2024, 2024.
- [59] 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.
- [60] Mark Chen. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.
- [61] Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan 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.
- [62] 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 τ 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 mbs (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
         $q_j \rightarrow Q$ 
         $p_j = \frac{1}{G} \sum_{i=1}^G \mathbb{I}(a_i = 1)$                                 ▷ Compute pass rate of  $q_j$ 
         $p_j \rightarrow P_a$ 

```

Step 2: Sort Q by pass rate

Sort Q in descending order of pass rate

Step 3: Compute the target size for completion

```

 $k = \lfloor \frac{|Q| + mbs - 1}{mbs} \rfloor$                                 ▷ Compute the smallest multiples of  $mbs$ 
 $T = k \times mbs$                                          ▷ 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
