KIMI K1.5: SCALING REINFORCEMENT LEARNING WITH LLMS

TECHNICAL REPORT OF KIMI K1.5

Kimi Team

ABSTRACT

Language model pretraining with next token prediction has proved effective for scaling compute but is limited to the amount of available training data. Scaling reinforcement learning (RL) unlocks a new axis for the continued improvement of artificial intelligence, with the promise that large language models (LLMs) can scale their training data by learning to explore with rewards. However, prior published work has not produced competitive results. In light of this, we report on the training practice of Kimi k1.5, our latest multi-modal LLM trained with RL, including its RL training techniques, multi-modal data recipes, and infrastructure optimization. Long context scaling and improved policy optimization methods are key ingredients of our approach, which establishes a simplistic, effective RL framework without relying on more complex techniques such as Monte Carlo tree search, value functions, and process reward models. Notably, our system achieves state-of-the-art reasoning performance across multiple benchmarks and modalities—e.g., 77.5 on AIME, 96.2 on MATH 500, 94-th percentile on Codeforces, 74.9 on MathVista—matching OpenAI's o1. Moreover, we present effective long2short methods that use long-CoT techniques to improve short-CoT models, yielding state-of-the-art short-CoT reasoning results—e.g., 60.8 on AIME, 94.6 on MATH500, 47.3 on LiveCodeBench—outperforming existing short-CoT models such as GPT-4o and Claude Sonnet 3.5 by a large margin (up to +550%). The models will be available soon.

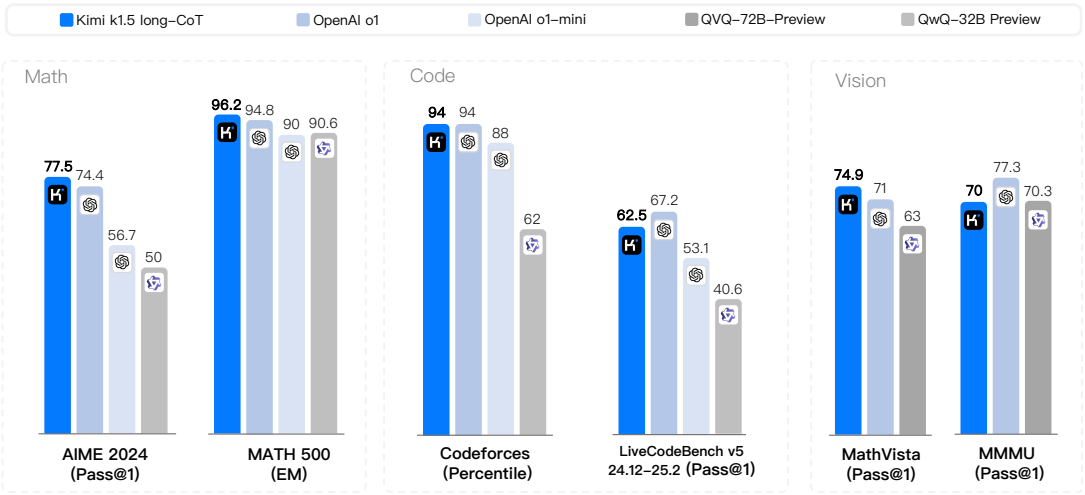


Figure 1: Kimi k1.5 long-CoT results.

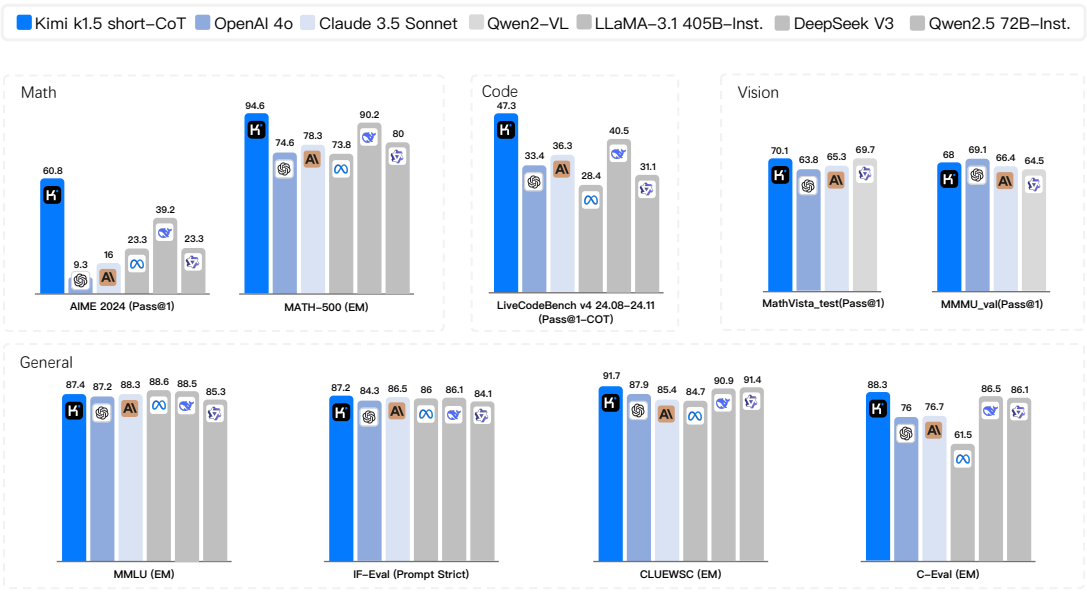


Figure 2: Kimi k1.5 short-CoT results.

1 Introduction

Language model pretraining with next token prediction has been studied under the context of the scaling law, where proportionally scaling model parameters and data sizes leads to the continued improvement of intelligence. (Kaplan et al. 2020; Hoffmann et al. 2022) However, this approach is limited to the amount of available high-quality training data (Villalobos et al. 2024; Muennighoff et al. 2023). In this report, we present the training recipe of Kimi k1.5, our latest multi-modal LLM trained with reinforcement learning (RL). The goal is to explore a possible new axis for continued scaling. Using RL with LLMs, the models learns to explore with rewards and thus is not limited to a pre-existing static dataset.

There are a few key ingredients about the design and training of k1.5.

- **Long context scaling**. We scale the context window of RL to 128k and observe continued improvement of performance with an increased context length. A key idea behind our approach is to use partial rollouts to improve training efficiency—i.e., sampling new trajectories by reusing a large chunk of previous trajectories, avoiding the cost to re-generate the new trajectories from scratch. Our observation identifies the context length as a key dimension of the continued scaling of RL with LLMs.
- **Improved policy optimization**. We derive a formulation of RL with long-CoT and employ a variant of online mirror descent for robust policy optimization. This algorithm is further improved by our effective sampling strategy, length penalty, and optimization of the data recipe.
- **Simplistic Framework**. Long context scaling, combined with the improved policy optimization methods, establishes a simplistic RL framework for learning with LLMs. Since we are able to scale the context length, the learned CoTs exhibit the properties of planning, reflection, and correction. An increased context length has an effect of increasing the number of search steps. As a result, we show that strong performance can be achieved without relying on more complex techniques such as Monte Carlo tree search, value functions, and process reward models.
- **Mutimodalities**. Our model is jointly trained on text and vision data, which has the capabilities of jointly reasoning over the two modalities.

Moreover, we present effective long2short methods that use long-CoT techniques to improve short-CoT models. Specifically, our approaches include applying length penalty with long-CoT activations and model merging.

Our long-CoT version achieves state-of-the-art reasoning performance across multiple benchmarks and modalities—e.g., 77.5 on AIME, 96.2 on MATH 500, 94-th percentile on Codeforces, 74.9 on MathVista—matching OpenAI's o1. Our model also achieves state-of-the-art short-CoT reasoning results—e.g., 60.8 on AIME, 94.6 on MATH500, 47.3 on LiveCodeBench—outperforming existing short-CoT models such as GPT-4o and Claude Sonnet 3.5 by a large margin (up to +550%). Results are shown in Figures 1 and 2.

2 Approach: Reinforcement Learning with LLMs

The development of Kimi k1.5 consists of several stages: pretraining, vanilla supervised fine-tuning (SFT), long-CoT supervised fine-turning, and reinforcement learning (RL). This report focuses on RL, beginning with an overview of the RL prompt set curation (Section 2.1) and long-CoT supervised finetuning (Section 2.2), followed by an in-depth discussion of RL training strategies in Section 2.3. Additional details on pretraining and vanilla supervised finetuning can be found in Section 2.5.

2.1 RL Prompt Set Curation

Through our preliminary experiments, we found that the quality and diversity of the RL prompt set play a critical role in ensuring the effectiveness of reinforcement learning. A well-constructed prompt set not only guides the model toward robust reasoning but also mitigates the risk of reward hacking and overfitting to superficial patterns. Specifically, three key properties define a high-quality RL prompt set:

- **Diverse Coverage**: Prompts should span a wide array of disciplines, such as STEM, coding, and general reasoning, to enhance the model's adaptability and ensure broad applicability across different domains.
- **Balanced Difficulty**: The prompt set should include a well-distributed range of easy, moderate, and difficult questions to facilitate gradual learning and prevent overfitting to specific complexity levels.
- **Accurate Evaluability**: Prompts should allow objective and reliable assessment by verifiers, ensuring that model performance is measured based on correct reasoning rather than superficial patterns or random guess.

To achieve diverse coverage in the prompt set, we employ automatic filters to select questions that require rich reasoning and are straightforward to evaluate. Our dataset includes problems from various domains, such as STEM fields, competitions, and general reasoning tasks, incorporating both text-only and image-text question-answering data. Furthermore, we developed a tagging system to categorize prompts by domain and discipline, ensuring balanced representation across different subject areas (M. Li et al. 2023; W. Liu et al. 2023).

We adopt a model-based approach that leverages the model's own capacity to adaptively assess the difficulty of each prompt. Specifically, for every prompt, an SFT model generates answers ten times using a relatively high sampling temperature. The pass rate is then calculated and used as a proxy for the prompt's difficulty—the lower the pass rate, the higher the difficulty. This approach allows difficulty evaluation to be aligned with the model's intrinsic capabilities, making it highly effective for RL training. By leveraging this method, we can prefilter most trivial cases and easily explore different sampling strategies during RL training.

To avoid potential reward hacking (Everitt et al. 2021; Pan et al. 2022), we need to ensure that both the reasoning process and the final answer of each prompt can be accurately verified. Empirical observations reveal that some complex reasoning problems may have relatively simple and easily guessable answers, leading to false positive verification—where the model reaches the correct answer through an incorrect reasoning process. To address this issue, we exclude questions that are prone to such errors, such as multiple-choice, true/false, and proof-based questions. Furthermore, for general question-answering tasks, we propose a simple yet effective method to identify and remove easy-to-hack prompts. Specifically, we prompt a model to guess potential answers without any CoT reasoning steps. If the model predicts the correct answer within N attempts, the prompt is considered too easy-to-hack and removed. We found that setting $N = 8$ can remove the majority easy-to-hack prompts. Developing more advanced verification models remains an open direction for future research.

2.2 Long-CoT Supervised Fine-Tuning

With the refined RL prompt set, we employ prompt engineering to construct a small yet high-quality long-CoT warmup dataset, containing accurately verified reasoning paths for both text and image inputs. This approach resembles rejection sampling (RS) but focuses on generating long-CoT reasoning paths through prompt engineering. The resulting warmup dataset is designed to encapsulate key cognitive processes that are fundamental to human-like reasoning, such as **planning**, where the model systematically outlines steps before execution; **evaluation**, involving critical assessment of intermediate steps; **reflection**, enabling the model to reconsider and refine its approach; and **exploration**, encouraging consideration of alternative solutions. By performing a lightweight SFT on this warm-up dataset, we effectively prime the model to internalize these reasoning strategies. As a result, the fine-tuned long-CoT model demonstrates improved capability in generating more detailed and logically coherent responses, which enhances its performance across diverse reasoning tasks.

2.3 Reinforcement Learning

2.3.1 Problem Setting

Given a training dataset $\mathcal{D} = \{(x_i, y_i^*)\}_{i=1}^n$ of problems x_i and corresponding ground truth answers y_i^*, our goal is to train a policy model π_θ to accurately solve test problems. In the context of complex reasoning, the mapping of problem x to solution y is non-trivial. To tackle this challenge, the *chain of thought* (CoT) method proposes to use a sequence of intermediate steps $z = (z_1, z_2, \ldots, z_m)$ to bridge x and y, where each z_i is a coherent sequence of tokens that acts as a significant intermediate step toward solving the problem (J. Wei et al. 2022). When solving problem x, thoughts $z_t \sim \pi_\theta(\cdot|x, z_1, \ldots, z_{t-1})$ are auto-regressively sampled, followed by the final answer $y \sim \pi_\theta(\cdot|x, z_1, \ldots, z_m)$. We use $y, z \sim \pi_\theta$ to denote this sampling procedure. Note that both the thoughts and final answer are sampled as a language sequence.

To further enhance the model's reasoning capabilities, *planning* algorithms are employed to explore various thought processes, generating improved CoT at inference time (Yao et al. 2024; Y. Wu et al. 2024; Snell et al. 2024). The core insight of these approaches is the explicit construction of a search tree of thoughts guided by value estimations. This allows the model to explore diverse continuations of a thought process or backtrack to investigate new directions when encountering dead ends. In more detail, let $\mathcal{T}$ be a search tree where each node represents a partial solution $s = (x, z_{1:|s|})$. Here s consists of the problem x and a sequence of thoughts $z_{1:|s|} = (z_1, \ldots, z_{|s|})$ leading up to that node, with $|s|$ denoting number of thoughts in the sequence. The planning algorithm uses a critic model v to provide feedback $v(x, z_{1:|s|})$, which helps evaluate the current progress towards solving the problem and identify any errors in the existing partial solution. We note that the feedback can be provided by either a discriminative score or a language sequence(L. Zhang et al. 2024). Guided by the feedbacks for all $s \in \mathcal{T}$, the planning algorithm selects the most promising node for expansion, thereby growing the search tree. The above process repeats iteratively until a full solution is derived.

We can also approach planning algorithms from an *algorithmic perspective*. Given past search history available at the t-th iteration $(s_1, v(s_1), \ldots, s_{t-1}, v(s_{t-1}))$, a planning algorithm $\mathcal{A}$ iteratively determines the next search direction $\mathcal{A}(s_t|s_1, v(s_1), \ldots, s_{t-1}, v(s_{t-1}))$ and provides feedbacks for the current search progress $\mathcal{A}(v(s_t)|s_1, v(s_1), \ldots, s_t)$. Since both thoughts and feedbacks can be viewed as intermediate reasoning steps, and these components can both be represented as sequence of language tokens, we use z to replace s and v to simplify the notations. Accordingly, we view a planning algorithm as a mapping that directly acts on a sequence of reasoning steps $\mathcal{A}(\cdot|z_1, z_2, \ldots)$. In this framework, all information stored in the search tree used by the planning algorithm is flattened into the full context provided to the algorithm. This provides an intriguing perspective on generating high-quality CoT: Rather than explicitly constructing a search tree and implementing a planning algorithm, we could potentially train a model to approximate this process. Here, the number of thoughts (i.e., language tokens) serves as an analogy to the computational budget traditionally allocated to planning algorithms. Recent advancements in long context windows facilitate seamless scalability during both the training and testing phases. If feasible, this method enables the model to run an implicit search over the reasoning space directly via auto-regressive predictions. Consequently, the model not only learns to solve a set of training problems but also develops the ability to tackle individual problems effectively, leading to improved generalization to unseen test problems.

We thus consider training the model to generate CoT with reinforcement learning (RL) (OpenAI 2024). Let r be a reward model that justifies the correctness of the proposed answer y for the given problem x based on the ground truth y^*, by assigning a value $r(x, y, y^*) \in \{0, 1\}$. For verifiable problems, the reward is directly determined by predefined criteria or rules. For example, in coding problems, we assess whether the answer passes the test cases. For problems with free-form ground truth, we train a reward model $r(x, y, y^*)$ that predicts if the answer matches the ground truth. Given a problem x, the model π_θ generates a CoT and the final answer through the sampling procedure $z \sim \pi_\theta(\cdot|x)$, $y \sim \pi_\theta(\cdot|x, z)$. The quality of the generated CoT is evaluated by whether it can lead to a correct final answer. In summary, we consider the following objective to optimize the policy

$$\max_\theta \mathbb{E}_{(x,y^*)\sim\mathcal{D},(y,z)\sim\pi_\theta} \left[r(x, y, y^*)\right] . \tag{1}$$

By scaling up RL training, we aim to train a model that harnesses the strengths of both simple prompt-based CoT and planning-augmented CoT. The model still auto-regressively sample language sequence during inference, thereby circumventing the need for the complex parallelization required by advanced planning algorithms during deployment. However, a key distinction from simple prompt-based methods is that the model should not merely follow a series of reasoning steps. Instead, it should also learn critical planning skills including error identification, backtracking and solution refinement by leveraging the entire set of explored thoughts as contextual information.

2.3.2 Policy Optimization

We apply a variant of online policy mirror decent as our training algorithm (Abbasi-Yadkori et al. 2019; Mei et al. 2019; Tomar et al. 2020). The algorithm performs iteratively. At the i-th iteration, we use the current model π_{θ_i} as a reference model and optimize the following relative entropy regularized policy optimization problem,

$$\max_{\theta} \mathbb{E}_{(x,y^*)\sim\mathcal{D}} \left[\mathbb{E}_{(y,z)\sim\pi_\theta} \left[r(x,y,y^*) \right] - \tau \mathrm{KL}(\pi_\theta(x) || \pi_{\theta_i}(x)) \right] , \tag{2}$$

where $\tau > 0$ is a parameter controlling the degree of regularization. This objective has a closed form solution

$$\pi^*(y,z|x) = \pi_{\theta_i}(y,z|x) \exp(r(x,y,y^*)/\tau)/Z \, .$$

Here $Z = \sum_{y',z'} \pi_{\theta_i}(y',z'|x) \exp(r(x,y',y^*)/\tau)$ is the normalization factor. Taking logarithm of both sides we have for *any* (y,z) the following constraint is satisfied, which allows us to leverage off-policy data during optimization

$$r(x,y,y^*) - \tau \log Z = \tau \log \frac{\pi^*(y,z|x)}{\pi_{\theta_i}(y,z|x)} \, .$$

This motivates the following surrogate loss

$$L(\theta) = \mathbb{E}_{(x,y^*)\sim\mathcal{D}} \left[\mathbb{E}_{(y,z)\sim\pi_{\theta_i}} \left[\left(r(x,y,y^*) - \tau \log Z - \tau \log \frac{\pi_\theta(y,z|x)}{\pi_{\theta_i}(y,z|x)} \right)^2 \right] \right] .$$

To approximate $\tau \log Z$, we use samples $(y_1,z_1),\ldots,(y_k,z_k) \sim \pi_{\theta_i}$: $\tau \log Z \approx \tau \log \frac{1}{k} \sum_{j=1}^{k} \exp(r(x,y_j,y^*)/\tau)$. We also find that using empirical mean of sampled rewards $\overline{r} = \mathrm{mean}(r(x,y_1,y^*),\ldots,r(x,y_k,y^*))$ yields effective practical results. This is reasonable since $\tau \log Z$ approaches the expected reward under π_{θ_i} as $\tau \to \infty$. Finally, we conclude our learning algorithm by taking the gradient of surrogate loss. For each problem x, k responses are sampled using the reference policy π_{θ_i}, and the gradient is given by

$$\frac{1}{k} \sum_{j=1}^{k} \left(\nabla_\theta \log \pi_\theta(y_j,z_j|x)(r(x,y_j,y^*) - \overline{r}) - \frac{\tau}{2} \nabla_\theta \left(\log \frac{\pi_\theta(y_j,z_j|x)}{\pi_{\theta_i}(y_j,z_j|x)} \right)^2 \right) , \tag{3}$$

To those familiar with policy gradient methods, this gradient resembles the policy gradient of (2) using the mean of sampled rewards as the baseline (Kool et al. 2019; Ahmadian et al. 2024). The main differences are that the responses are sampled from π_{θ_i} rather than on-policy, and an l_2-regularization is applied. Thus we could see this as the natural extension of a usual on-policy regularized policy gradient algorithm to the off-policy case (Nachum et al. 2017). We sample a batch of problems from $\mathcal{D}$ and update the parameters to θ_{i+1}, which subsequently serves as the reference policy for the next iteration. Since each iteration considers a different optimization problem due to the changing reference policy, we also reset the optimizer at the start of each iteration.

We exclude the value network in our training system which has also been exploited in previous studies (Ahmadian et al. 2024). While this design choice significantly improves training efficiency, we also hypothesize that the conventional use of value functions for credit assignment in classical RL may not be suitable for our context. Consider a scenario where the model has generated a partial CoT $(z_1, z_2, \ldots, z_t)$ and there are two potential next reasoning steps: z_{t+1} and z'_{t+1}. Assume that z_{t+1} directly leads to the correct answer, while z'_{t+1} contains some errors. If an oracle value function were accessible, it would indicate that z_{t+1} preserves a higher value compared to z'_{t+1}. According to the standard credit assignment principle, selecting z'_{t+1} would be penalized as it has a negative advantages relative to the current policy. However, exploring z'_{t+1} is extremely valuable for training the model to generate long CoT. By using the justification of the final answer derived from a long CoT as the reward signal, the model can learn the pattern of trial and error from taking z'_{t+1} as long as it successfully recovers and reaches the correct answer. The key takeaway from this example is that we should encourage the model to explore diverse reasoning paths to enhance its capability in solving complex problems. This exploratory approach generates a wealth of experience that supports the development of critical planning skills. Our primary goal is not confined to attaining high accuracy on training problems but focuses on equipping the model with effective problem-solving strategies, ultimately improving its performance on test problems.

2.3.3 Length Penalty

We observe an overthinking phenomenon that the model's response length significantly increases during RL training. Although this leads to better performance, an excessively lengthy reasoning process is costly during training and inference, and overthinking is often not preferred by humans. To address this issue, we introduce a length reward to restrain the rapid growth of token length, thereby improving the model's token efficiency. Given k sampled responses

$(y_1, z_1), \ldots, (y_k, z_k)$ of problem x with true answer y^*, let $\text{len}(i)$ be the length of (y_i, z_i), $\text{min\_len} = \min_i \text{len}(i)$ and $\text{max\_len} = \max_i \text{len}(i)$. If $\text{max\_len} = \text{min\_len}$, we set length reward zero for all responses, as they have the same length. Otherwise the length reward is given by

$$\text{len\_reward(i)} = \begin{cases} \lambda & \text{If } r(x, y_i, y^*) = 1 \\ \min(0, \lambda) & \text{If } r(x, y_i, y^*) = 0 \end{cases}, \quad \text{where } \lambda = 0.5 - \frac{\text{len}(i) - \text{min\_len}}{\text{max\_len} - \text{min\_len}}.$$

In essence, we promote shorter responses and penalize longer responses among correct ones, while explicitly penalizing long responses with incorrect answers. This length-based reward is then added to the original reward with a weighting parameter.

In our preliminary experiments, length penalty may slow down training during the initial phases. To alleviate this issue, we propose to gradually warm up the length penalty during training. Specifically, we employ standard policy optimization without length penalty, followed by a constant length penalty for the rest of training.

2.3.4 Sampling Strategies

Although RL algorithms themselves have relatively good sampling properties (with more difficult problems providing larger gradients), their training efficiency is limited. Consequently, some well-defined prior sampling methods can yield potentially greater performance gains. We exploit multiple signals to further improve the sampling strategy. First, the RL training data we collect naturally come with different difficulty labels. For example, a math competition problem is more difficult than a primary school math problem. Second, because the RL training process samples the same problem multiple times, we can also track the success rate for each individual problem as a metric of difficulty. We propose two sampling methods to utilize these priors to improve training efficiency.

Curriculum Sampling We start by training on easier tasks and gradually progress to more challenging ones. Since the initial RL model has limited performance, spending a restricted computation budget on very hard problems often yields few correct samples, resulting in lower training efficiency. Meanwhile, our collected data naturally includes grade and difficulty labels, making difficulty-based sampling an intuitive and effective way to improve training efficiency.

Prioritized Sampling In addition to curriculum sampling, we use a prioritized sampling strategy to focus on problems where the model underperforms. We track the success rates s_i for each problem i and sample problems proportional to $1 - s_i$, so that problems with lower success rates receive higher sampling probabilities. This directs the model's efforts toward its weakest areas, leading to faster learning and better overall performance.

2.3.5 More Details on Training Recipe

Test Case Generation for Coding Since test cases are not available for many coding problems from the web, we design a method to automatically generate test cases that serve as a reward to train our model with RL. Our focus is primarily on problems that do not require a special judge. We also assume that ground truth solutions are available for these problems so that we can leverage the solutions to generate higher quality test cases.

We utilize the widely recognized test case generation library, CYaRon[1], to enhance our approach. We employ our base Kimi k1.5 to generate test cases based on problem statements. The usage statement of CYaRon and the problem description are provided as the input to the generator. For each problem, we first use the generator to produce 50 test cases and also randomly sample 10 ground truth submissions for each test case. We run the test cases against the submissions. A test case is deemed valid if at least 7 out of 10 submissions yield matching results. After this round of filtering, we obtain a set of selected test cases. A problem and its associated selected test cases are added to our training set if at least 9 out of 10 submissions pass the entire set of selected test cases.

In terms of statistics, from a sample of 1,000 online contest problems, approximately 614 do not require a special judge. We developed 463 test case generators that produced at least 40 valid test cases, leading to the inclusion of 323 problems in our training set.

Reward Modeling for Math One challenge in evaluating math solutions is that different written forms can represent the same underlying answer. For instance, $a^2 - 4$ and $(a + 2)(a - 2)$ may both be valid solutions to the same problem. We adopted two methods to improve the reward model's scoring accuracy:

1. Classic RM: Drawing inspiration from the InstructGPT (Ouyang et al. 2022) methodology, we implemented a value-head based reward model and collected approximately 800k data points for fine-tuning. The model ultimately

[1] https://github.com/luogu-dev/cyaron

takes as input the "question," the "reference answer," and the "response," and outputs a single scalar that indicates whether the response is correct.

2. Chain-of-Thought RM: Recent research (Ankner et al. 2024; McAleese et al. 2024) suggests that reward models augmented with chain-of-thought (CoT) reasoning can significantly outperform classic approaches, particularly on tasks where nuanced correctness criteria matter—such as mathematics. Therefore, we collected an equally large dataset of about 800k CoT-labeled examples to fine-tune the Kimi model. Building on the same inputs as the Classic RM, the chain-of-thought approach explicitly generates a step-by-step reasoning process before providing a final correctness judgment in JSON format, enabling more robust and interpretable reward signals.

During our manual spot checks, the Classic RM achieved an accuracy of approximately **84.4**, while the Chain-of-Thought RM reached **98.5** accuracy. In the RL training process, we adopted the Chain-of-Thought RM to ensure more correct feedback.

Vision Data To improve the model's real-world image reasoning capabilities and to achieve a more effective alignment between visual inputs and large language models (LLMs), our vision reinforcement learning (Vision RL) data is primarily sourced from three distinct categories: Real-world data, Synthetic visual reasoning data, and Text-rendered data.

1. The real-world data encompass a range of science questions across various grade levels that require graphical comprehension and reasoning, location guessing tasks that necessitate visual perception and inference, and data analysis that involves understanding complex charts, among other types of data. These datasets improve the model's ability to perform visual reasoning in real-world scenarios.
2. Synthetic visual reasoning data is artificially generated, including procedurally created images and scenes aimed at improving specific visual reasoning skills, such as understanding spatial relationships, geometric patterns, and object interactions. These synthetic datasets offer a controlled environment for testing the model's visual reasoning capabilities and provide an endless supply of training examples.
3. Text-rendered data is created by converting textual content into visual format, enabling the model to maintain consistency when handling text-based queries across different modalities. By transforming text documents, code snippets, and structured data into images, we ensure the model provides consistent responses regardless of whether the input is pure text or text rendered as images (like screenshots or photos). This also helps to enhance the model's capability when dealing with text-heavy images.

Each type of data is essential in building a comprehensive visual language model that can effectively manage a wide range of real-world applications while ensuring consistent performance across various input modalities.

2.4 Long2short: Context Compression for Short-CoT Models

Though long-CoT models achieve strong performance, it consumes more test-time tokens compared to standard short-CoT LLMs. However, it is possible to transfer the thinking priors from long-CoT models to short-CoT models so that performance can be improved even with limited test-time token budgets. We present several approaches for this long2short problem, including model merging (Yang et al. 2024), shortest rejection sampling, DPO (Rafailov et al. 2024), and long2short RL. Detailed descriptions of these methods are provided below:

Model Merging Model merging has been found to be useful in maintaining generalization ability. We also discovered its effectiveness in improving token efficiency when merging a long-cot model and a short-cot model. This approach combines a long-cot model with a shorter model to obtain a new one without training. Specifically, we merge the two models by simply averaging their weights.

Shortest Rejection Sampling We observed that our model generates responses with a large length variation for the same problem. Based on this, we designed the Shortest Rejection Sampling method. This method samples the same question n times (in our experiments, $n = 8$) and selects the shortest correct response for supervised fine-tuning.

DPO Similar with Shortest Rejection Sampling, we utilize the Long CoT model to generate multiple response samples. The shortest correct solution is selected as the positive sample, while longer responses are treated as negative samples, including both wrong longer responses and correct longer responses (1.5 times longer than the chosen positive sample). These positive-negative pairs form the pairwise preference data used for DPO training.

Long2short RL After a standard RL training phase, we select a model that offers the best balance between performance and token efficiency to serve as the base model, and conduct a separate long2short RL training phase. In this second phase, we apply the length penalty introduced in Section 2.3.3, and significantly reduce the maximum rollout length to further penalize responses that exceed the desired length while possibly correct.

2.5 Other Training Details

2.5.1 Pretraining

The Kimi k1.5 base model is trained on a diverse, high-quality multimodal corpus. The language data covers five domains: English, Chinese, Code, Mathematics Reasoning, and Knowledge. Multimodal data, including Captioning, Image-text Interleaving, OCR, Knowledge, and QA datasets, enables our model to acquire vision-language capabilities. Rigorous quality control ensures relevance, diversity, and balance in the overall pretrain dataset. Our pretraining proceeds in three stages: (1) Vision-language pretraining, where a strong language foundation is established, followed by gradual multimodal integration; (2) Cooldown, which consolidates capabilities using curated and synthetic data, particularly for reasoning and knowledge-based tasks; and (3) Long-context activation, extending sequence processing to 131,072 tokens. More details regarding our pretraining efforts can be found in Appendix C.

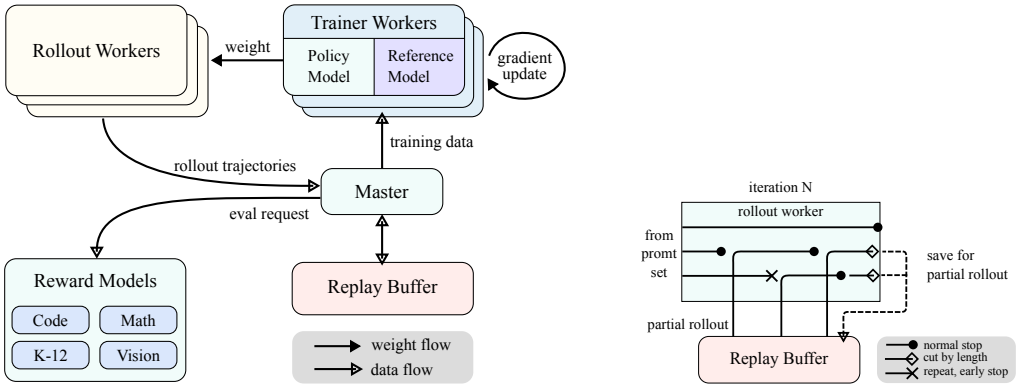
2.5.2 Vanilla Supervised Finetuning

We create the vanilla SFT corpus covering multiple domains. For non-reasoning tasks, including question-answering, writing, and text processing, we initially construct a seed dataset through human annotation. This seed dataset is used to train a seed model. Subsequently, we collect a diverse of prompts and employ the seed model to generate multiple responses to each prompt. Annotators then rank these responses and refine the top-ranked response to produce the final version. For reasoning tasks such as math and coding problems, where rule-based and reward modeling based verifications are more accurate and efficient than human judgment, we utilize rejection sampling to expand the SFT dataset.

Our vanilla SFT dataset comprises approximately 1 million text examples. Specifically, 500k examples are for general question answering, 200k for coding, 200k for math and science, 5k for creative writing, and 20k for long-context tasks such as summarization, doc-qa, translation, and writing. In addition, we construct 1 million text-vision examples encompassing various categories including chart interpretation, OCR, image-grounded conversations, visual coding, visual reasoning, and math/science problems with visual aids.

We first train the model at the sequence length of 32k tokens for 1 epoch, followed by another epoch at the sequence length of 128k tokens. In the first stage (32k), the learning rate decays from 2×10^{-5} to 2×10^{-6}, before it re-warmups to 1×10^{-5} in the second stage (128k) and finally decays to 1×10^{-6}. To improve training efficiency, we pack multiple training examples into each single training sequence.

2.6 RL Infrastructure



(a) System overview (b) Partial Rollout

Figure 3: Large Scale Reinforcement Learning Training System for LLM

2.6.1 Large Scale Reinforcement Learning Training System for LLM

In the realm of artificial intelligence, reinforcement learning (RL) has emerged as a pivotal training methodology for large language models (LLMs)(Ouyang et al. 2022)(Jaech et al. 2024), drawing inspiration from its success in mastering complex games like Go, StarCraft II, and Dota 2 through systems such as AlphaGo(Silver et al. 2017), AlphaStar(Vinyals et al. 2019), and OpenAI Dota Five (Berner et al. 2019). Following in this tradition, the Kimi k1.5 system adopts an iterative synchronous RL framework, meticulously designed to bolster the model's reasoning capabilities through persistent learning and adaptation. A key innovation in this system is the introduction of a Partial Rollout technique, designed to optimize the handling of complex reasoning trajectories.

The RL training system as illustrated in Figure 3a operates through an iterative synchronous approach, with each iteration encompassing a rollout phase and a training phase. During the rollout phase, rollout workers, coordinated by a central master, generate rollout trajectories by interacting with the model, producing sequences of responses to various inputs. These trajectories are then stored in a replay buffer, which ensures a diverse and unbiased dataset for training by disrupting temporal correlations. In the subsequent training phase, trainer workers access these experiences to update the model's weights. This cyclical process allows the model to continuously learn from its actions, adjusting its strategies over time to enhance performance.

The central master serves as the central conductor, managing the flow of data and communication between the rollout workers, trainer workers, evaluation with reward models and the replay buffer. It ensures that the system operates harmoniously, balancing the load and facilitating efficient data processing.

The trainer workers access these rollout trajectories, whether completed in a single iteration or divided across multiple iterations, to compute gradient updates that refine the model's parameters and enhance its performance. This process is overseen by a reward model, which evaluates the quality of the model's outputs and provides essential feedback to guide the training process. The reward model's evaluations are particularly pivotal in determining the effectiveness of the model's strategies and steering the model towards optimal performance.

Moreover, the system incorporates a code execution service, which is specifically designed to handle code-related problems and is integral to the reward model. This service evaluates the model's outputs in practical coding scenarios, ensuring that the model's learning is closely aligned with real-world programming challenges. By validating the model's solutions against actual code executions, this feedback loop becomes essential for refining the model's strategies and enhancing its performance in code-related tasks.

2.6.2 Partial Rollouts for Long CoT RL

One of the primary ideas of our work is to scale long-context RL training. Partial rollouts is a key technique that effectively addresses the challenge of handling long-CoT features by managing the rollouts of both long and short trajectories. This technique establishes a fixed output token budget, capping the length of each rollout trajectory. If a trajectory exceeds the token limit during the rollout phase, the unfinished portion is saved to the replay buffer and continued in the next iteration. It ensures that no single lengthy trajectory monopolizes the system's resources. Moreover, since the rollout workers operate asynchronously, when some are engaged with long trajectories, others can independently process new, shorter rollout tasks. The asynchronous operation maximizes computational efficiency by ensuring that all rollout workers are actively contributing to the training process, thereby optimizing the overall performance of the system.

As illustrated in Figure 3b, the partial rollout system works by breaking down long responses into segments across iterations (from iter n-m to iter n). The Replay Buffer acts as a central storage mechanism that maintains these response segments, where only the current iteration (iter n) requires on-policy computation. Previous segments (iter n-m to n-1) can be efficiently reused from the buffer, eliminating the need for repeated rollouts. This segmented approach significantly reduces the computational overhead: instead of rolling out the entire response at once, the system processes and stores segments incrementally, allowing for the generation of much longer responses while maintaining fast iteration times. During training, certain segments can be excluded from loss computation to further optimize the learning process, making the entire system both efficient and scalable.

The implementation of partial rollouts also offers repeat detection. The system identifies repeated sequences in the generated content and terminates them early, reducing unnecessary computation while maintaining output quality. Detected repetitions can be assigned additional penalties, effectively discouraging redundant content generation in the prompt set.

2.6.3 Hybrid Deployment of Training and Inference

The RL training process comprises of the following phases:

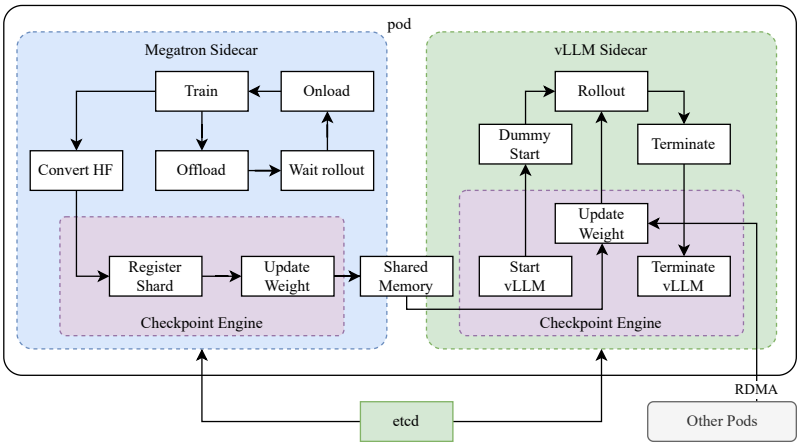


Figure 4: Hybrid Deployment Framework

- **Training Phase:** At the outset, Megatron (Shoeybi et al. 2020) and vLLM (Kwon et al. 2023) are executed within separate containers, encapsulated by a shim process known as checkpoint-engine (Section 2.6.3). Megatron commences the training procedure. After the training is completed, Megatron offloads the GPU memory and prepares to transfer current weights to vLLM.
- **Inference Phase:** Following Megatron's offloading, vLLM starts with dummy model weights and updates them with the latest ones transferred from Megatron via Mooncake (Qin et al. 2024). Upon completion of the rollout, the checkpoint-engine halts all vLLM processes.
- **Subsequent Training Phase:** Once the memory allocated to vLLM is released, Megatron onloads the memory and initiates another round of training.

We find existing works challenging to simultaneously support all the following characteristics.

- Complex parallelism strategy: Megatron may have different parallelism strategy with vLLM. Training weights distributing in several nodes in Megatron could be challenging to be shared with vLLM.
- Minimizing idle GPU resources: For On-Policy RL, recent works such as SGLang (L. Zheng et al. 2024) and vLLM might reserve some GPUs during the training process, which conversely could lead to idle training GPUs. It would be more efficient to share the same devices between training and inference.
- Capability of dynamic scaling: In some cases, a significant acceleration can be achieved by increasing the number of inference nodes while keeping the training process constant. Our system enables the efficient utilization of idle GPU nodes when needed.

As illustrated in Figure 4, we implement this hybrid deployment framework (Section 2.6.3) on top of Megatron and vLLM, achieving less than one minute from training to inference phase and about ten seconds conversely.

Hybrid Deployment Strategy We propose a hybrid deployment strategy for training and inference tasks, which leverages Kubernetes Sidecar containers sharing all available GPUs to collocate both workloads in one pod. The primary advantages of this strategy are:

- It facilitates efficient resource sharing and management, preventing train nodes idling while waiting for inference nodes when both are deployed on separate nodes.
- Leveraging distinct deployed images, training and inference can each iterate independently for better performance.
- The architecture is not limited to vLLM, other frameworks can be conveniently integrated.

Checkpoint Engine Checkpoint Engine is responsible for managing the lifecycle of the vLLM process, exposing HTTP APIs that enable triggering various operations on vLLM. For overall consistency and reliability, we utilize a global metadata system managed by the etcd service to broadcast operations and statuses.

It could be challenging to entirely release GPU memory by vLLM offloading primarily due to CUDA graphs, NCCL buffers and NVIDIA drivers. To minimize modifications to vLLM, we terminate and restart it when needed for better GPU utilization and fault tolerance.

The worker in Megatron converts the owned checkpoints into the Hugging Face format in shared memory. This conversion also takes Pipeline Parallelism and Expert Parallelism into account so that only Tensor Parallelism remains in these checkpoints. Checkpoints in shared memory are subsequently divided into shards and registered in the global metadata system. We employ Mooncake to transfer checkpoints between peer nodes over RDMA. Some modifications to vLLM are needed to load weight files and perform tensor parallelism conversion.

2.6.4 Code Sandbox

We developed the sandbox as a secure environment for executing user-submitted code, optimized for code execution and code benchmark evaluation. By dynamically switching container images, the sandbox supports different use cases through MultiPL-E (Cassano, Gouwar, D. Nguyen, S. Nguyen, et al. 2023), DMOJ Judge Server [2], Lean, Jupyter Notebook, and other images.

For RL in coding tasks, the sandbox ensures the reliability of training data judgment by providing consistent and repeatable evaluation mechanisms. Its feedback system supports multi-stage assessments, such as code execution feedback and repo-level editing, while maintaining a uniform context to ensure fair and equitable benchmark comparisons across programming languages.

We deploy the service on Kubernetes for scalability and resilience, exposing it through HTTP endpoints for external integration. Kubernetes features like automatic restarts and rolling updates ensure availability and fault tolerance.

To optimize performance and support RL environments, we incorporate several techniques into the code execution service to enhance efficiency, speed, and reliability. These include:

- **Using Crun:** We utilize `crun` as the container runtime instead of Docker, significantly reducing container startup times.
- **Cgroup Reusing:** We pre-create cgroups for container use, which is crucial in scenarios with high concurrency where creating and destroying cgroups for each container can become a bottleneck.
- **Disk Usage Optimization:** An overlay filesystem with an upper layer mounted as `tmpfs` is used to control disk writes, providing a fixed-size, high-speed storage space. This approach is beneficial for ephemeral workloads.

Method	Time (s)
Docker	0.12
Sandbox	0.04

(a) Container startup times

Method	Containers/sec
Docker	27
Sandbox	120

(b) Maximum containers started per second on a 16-core machine

These optimizations improve RL efficiency in code execution, providing a consistent and reliable environment for evaluating RL-generated code, essential for iterative training and model improvement.

3 Experiments

3.1 Evaluation

Since k1.5 is a multimodal model, we conducted comprehensive evaluation across various benchmarks for different modalities. The detailed evaluation setup can be found in Appendix D. Our benchmarks primarily consist of the following three categories:

- **Text Benchmark**: MMLU (Hendrycks et al. 2020), IF-Eval (J. Zhou et al. 2023), CLUEWSC (L. Xu et al. 2020), C-EVAL (Y. Huang et al. 2023)
- **Reasoning Benchmark**: HumanEval-Mul, LiveCodeBench (Jain et al. 2024), Codeforces, AIME 2024, MATH-500 (Lightman et al. 2023)
- **Vision Benchmark**: MMMU (Yue, Ni, et al. 2024), MATH-Vision (K. Wang et al. 2024), MathVista (Lu et al. 2023)

[2] https://github.com/DMOJ/judge-server

3.2 Main Results

K1.5 long-CoT model The performance of the Kimi k1.5 long-CoT model is presented in Table **??**. Through long-CoT supervised fine-tuning (described in Section 2.2) and vision-text joint reinforcement learning (discussed in Section 2.3), the model's long-term reasoning capabilities are enhanced significantly. The test-time computation scaling further strengthens its performance, enabling the model to achieve state-of-the-art results across a range of modalities. Our evaluation reveals marked improvements in the model's capacity to reason, comprehend, and synthesize information over extended contexts, representing a advancement in multi-modal AI capabilities.

K1.5 short-CoT model The performance of the Kimi k1.5 short-CoT model is presented in Table 3. This model integrates several techniques, including traditional supervised fine-tuning (discussed in Section 2.5.2), reinforcement learning (explored in Section 2.3), and long-to-short distillation (outlined in Section 2.4). The results demonstrate that the k1.5 short-CoT model delivers competitive or superior performance compared to leading open-source and proprietary models across multiple tasks. These include text, vision, and reasoning challenges, with notable strengths in natural language understanding, mathematics, coding, and logical reasoning.

		Language-only Model			Vision-Language Model			
	Benchmark (Metric)	**Qwen2.5 72B-Inst.**	**LLaMA-3.1 405B-Inst.**	**DeepSeek V3**	**Qwen2-VL**	**Claude-3.5-Sonnet-1022**	**GPT-4o 0513**	**Kimi k1.5**
Text	MMLU (EM)	85.3	**88.6**	88.5	-	88.3	87.2	87.4
	IF-Eval (Prompt Strict)	84.1	86	86.1	-	86.5	84.3	**87.2**
	CLUEWSC (EM)	91.4	84.7	90.9	-	85.4	87.9	**91.7**
	C-Eval (EM)	86.1	61.5	86.5	-	76.7	76.0	**88.3**
Reasoning	MATH-500 (EM)	80	73.8	90.2	-	78.3	74.6	**94.6**
	AIME 2024 (Pass@1)	23.3	23.3	39.2	-	16	9.3	**60.8**
	HumanEval-Mul (Pass@1)	77.3	77.2	**82.6**	-	81.7	80.5	81.5
	LiveCodeBench (Pass@1)	31.1	28.4	40.5	-	36.3	33.4	**47.3**
Vision	MathVista-Test (Pass@1)	-	-	-	69.7	65.3	63.8	**70.1**
	MMMU-Val (Pass@1)	-	-	-	64.5	66.4	**69.1**	68.0
	MathVision-Full (Pass@1)	-	-	-	26.6	**35.6**	30.4	31.0

Table 2: Performance of Kimi k1.5 short-CoT and flagship open-source and proprietary models. VLM model performance were obtained from the OpenCompass benchmark platform (https://opencompass.org.cn/).

		Language-only Model			Vision-Language Model			
	Benchmark (Metric)	**Qwen2.5 72B-Inst.**	**LLaMA-3.1 405B-Inst.**	**DeepSeek V3**	**Qwen2-VL**	**Claude-3.5-Sonnet-1022**	**GPT-4o 0513**	**Kimi k1.5**
Text	MMLU (EM)	85.3	**88.6**	88.5	-	88.3	87.2	87.4
	IF-Eval (Prompt Strict)	84.1	86	86.1	-	86.5	84.3	**87.2**
	CLUEWSC (EM)	91.4	84.7	90.9	-	85.4	87.9	**91.7**
	C-Eval (EM)	86.1	61.5	86.5	-	76.7	76.0	**88.3**
Reasoning	MATH-500 (EM)	80	73.8	90.2	-	78.3	74.6	**94.6**
	AIME 2024 (Pass@1)	23.3	23.3	39.2	-	16	9.3	**60.8**
	HumanEval-Mul (Pass@1)	77.3	77.2	**82.6**	-	81.7	80.5	81.5
	LiveCodeBench (Pass@1)	31.1	28.4	40.5	-	36.3	33.4	**47.3**
Vision	MathVista-Test (Pass@1)	-	-	-	69.7	65.3	63.8	**70.1**
	MMMU-Val (Pass@1)	-	-	-	64.5	66.4	**69.1**	68.0
	MathVision-Full (Pass@1)	-	-	-	26.6	**35.6**	30.4	31.0

Table 3: Performance of Kimi k1.5 short-CoT and flagship open-source and proprietary models. VLM model performance were obtained from the OpenCompass benchmark platform (https://opencompass.org.cn/).

3.3 Long Context Scaling

We employ a mid-sized model to study the scaling properties of RL with LLMs. Figure 5 illustrates the evolution of both training accuracy and response length across training iterations for the small model variant trained on the mathematical prompt set. As training progresses, we observe a concurrent increase in both response length and performance accuracy.

Notably, more challenging benchmarks exhibit a steeper increase in response length, suggesting that the model learns to generate more elaborate solutions for complex problems. Figure 6 indicates a strong correlation between the model's output context length and its problem-solving capabilities.

Our final run of k1.5 scales to 128k context length and observes continued improvedment on hard reasoning benchmarks such as Codeforces.

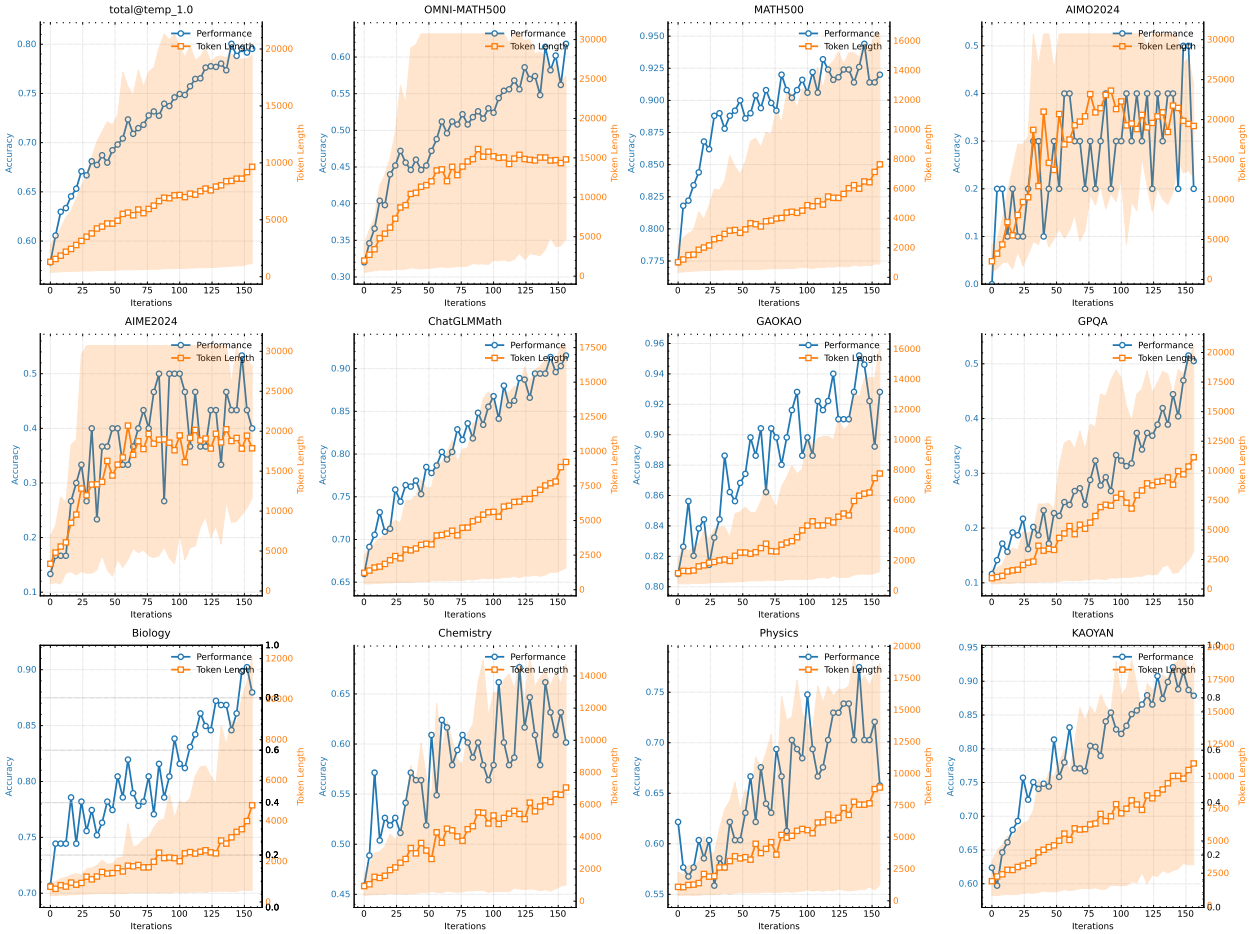


Figure 5: The changes on the training accuracy and length as train iterations grow. Note that the scores above come from an internal long-cot model with much smaller model size than k1.5 long-CoT model. The shaded area represents the 95% percentile of the response length.

3.4 Long2short

We compared the proposed long2short RL algorithm with the DPO, shortest rejection sampling, and model merge methods introduced in the Section 2.4, focusing on the token efficiency for the long2short problem (X. Chen et al. 2024), specifically how the obtained long-cot model can benefit a short model. In Figure 7, k1.5-long represents our long-cot model selected for long2short training. k1.5-short w/ rl refers to the short model obtained using the long2short RL training. k1.5-short w/ dpo denotes the short model with improved token efficiency through DPO training. k1.5-short w/ merge represents the model after model merging, while k1.5-short w/ merge + rs indicates the short model obtained by applying shortest rejection sampling to the merged model. k1.5-shortest represents the shortest model we obtained during the long2short training. As shown in Figure 7, the proposed long2short RL algorithm demonstrates the highest token efficiency compared other mehtods such as DPO and model merge. Notably, all models in the k1.5 series (marked in orange) demonstrate superior token efficiency compared to other models (marked in blue). For instance, k1.5-short w/ rl achieves a Pass@1 score of 60.8 on AIME2024 (averaged over 8 runs) while utilizing only 3,272 tokens on average. Similarly, k1.5-shortest attains a Pass@1 score of 88.2 on MATH500 while consuming approximately the same number of tokens as other short models.

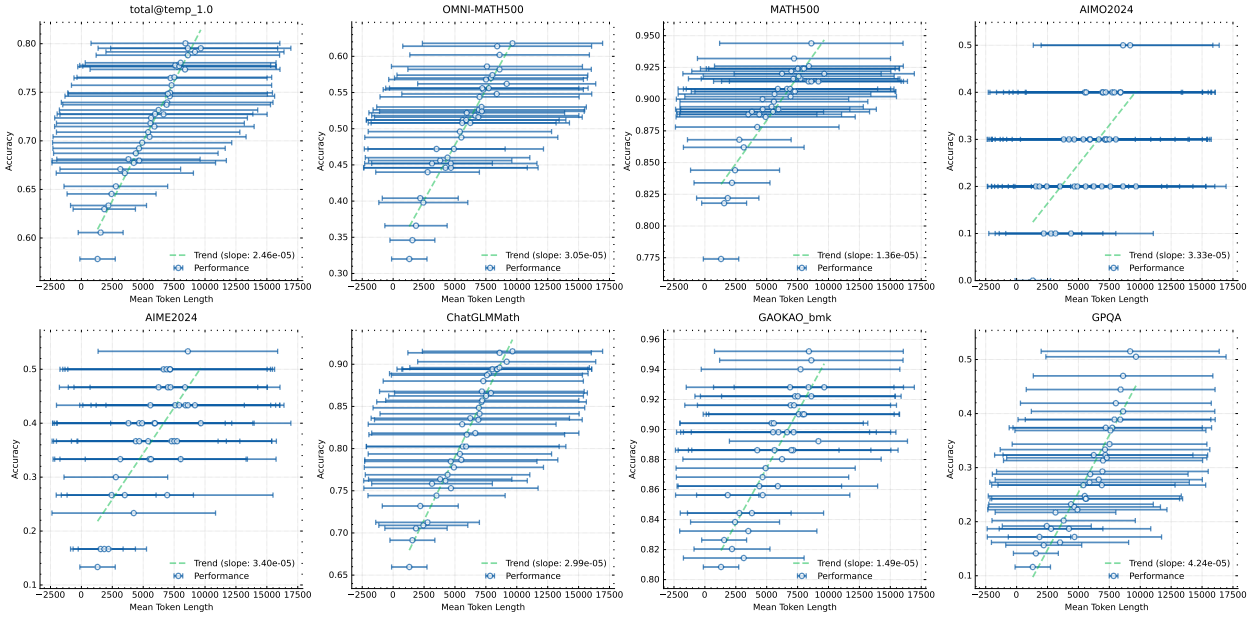


Figure 6: Model Performance Increases with Response Length

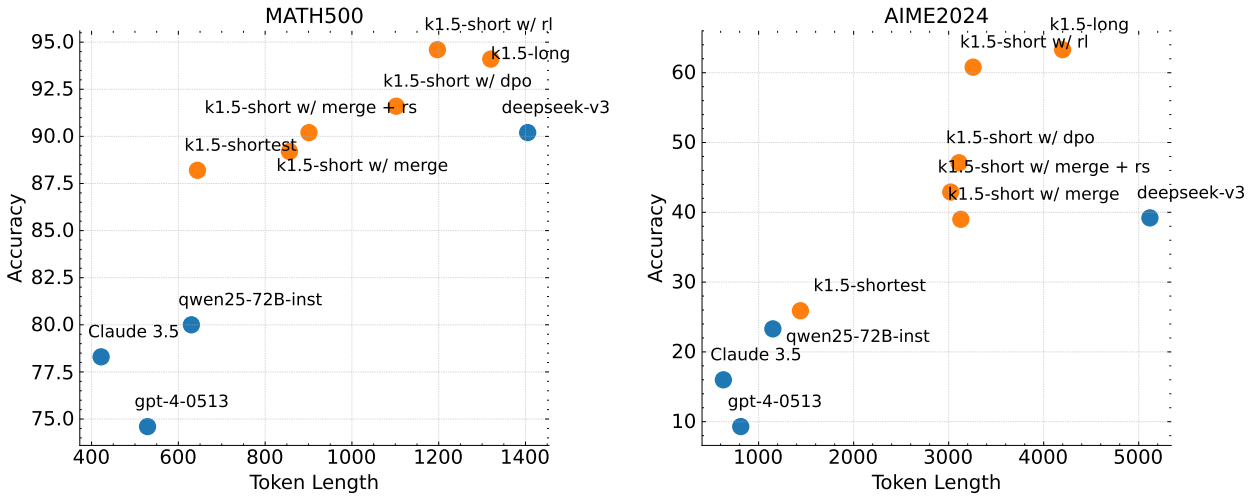


Figure 7: Long2Short Performance. All the k1.5 series demonstrate better token efficiency compared to other models.

3.5 Ablation Studies

Scaling of model size and context length Our main contribution is the application of RL to enhance the model's capacity for generating extended CoT, thereby improving its reasoning ability. A natural question arises: how does this compare to simply increasing the model size? To demonstrate the effectiveness of our approach, we trained two models of different sizes using the same dataset and recorded the evaluation results and average inference lengths from all checkpoints during RL training. These results are shown in Figure 8. Notably, although the larger model initially outperforms the smaller one, the smaller model can achieve comparable performance by utilizing longer CoTs optimized through RL. However, the larger model generally shows better token efficiency than the smaller model.

This indicates that if one targets the best possible performance, scaling the context length of a larger model has a higher upper bound and is more token efficient. However, if test-time compute has a budget, training smaller models with a larger context length may be viable solutions.

Effects of using negative gradients We investigate the effectiveness of using ReST (Gulcehre et al. 2023) as the policy optimization algorithm in our setting. The primary distinction between ReST and other RL-based methods including ours is that ReST iteratively refines the model by fitting the best response sampled from the current model, without applying negative gradients to penalize incorrect responses. As illustrated in Figure 10, our method exhibits superior sample complexity compared to ReST, indicating that the incorporation of negative gradients markedly enhances the model's efficiency in generating long CoT. Our method not only elevates the quality of reasoning but also optimizes the training process, achieving robust performance with fewer training samples. This finding suggests that the choice of policy optimization algorithm is crucial in our setting, as the performance gap between ReST and other RL-based methods is not as pronounced in other domains. Therefore, our results highlight the importance of selecting an appropriate optimization strategy to maximize effectiveness in generating long CoT.

Sampling strategies We further demonstrate the effectiveness of our curriculum sampling strategy, as introduced in Section 2.3.4. Our training dataset $\mathcal{D}$ comprises a diverse mix of problems with varying levels of difficulty. With our curriculum sampling method, we initially use $\mathcal{D}$ for a warm-up phase and then focus solely on hard questions to train the model. This approach is compared to a baseline method that employs a uniform sampling strategy without any curriculum adjustments. As illustrated in Figure 9, our results clearly show that the proposed curriculum sampling method significantly enhances the performance. This improvement can be attributed to the method's ability to progressively challenge the model, allowing it to develop a more robust understanding and competency in handling complex problems. By focusing training efforts on more difficult questions after an initial general introduction, the model can better strengthen its reasoning and problem solving capabilities.

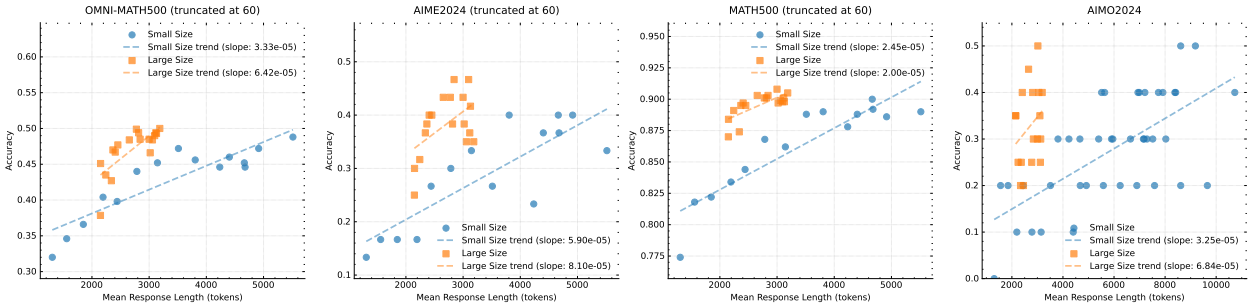


Figure 8: Model Performance vs Response Length of Different Model Sizes

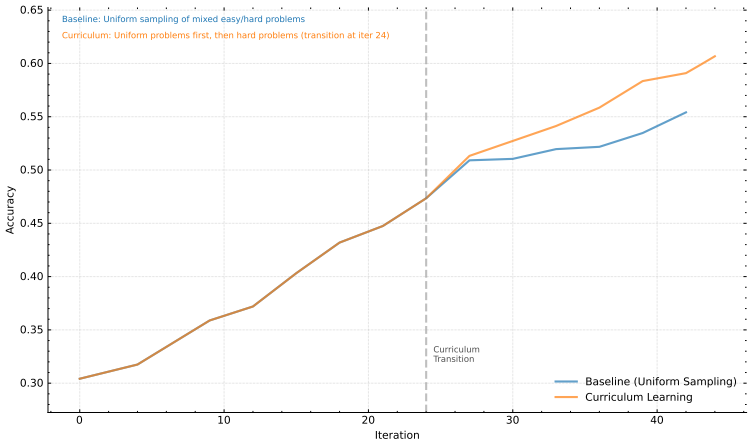


Figure 9: Analysis of curriculum learning approaches on model performance.

4 Conclusions

We present the training recipe and system design of k1.5, our latest multi-modal LLM trained with RL. One of the key insights we extract from our practice is that the scaling of context length is crucial to the continued improvement of LLMs. We employ optimized learning algorithms and infrastructure optimization such as partial rollouts to achieve

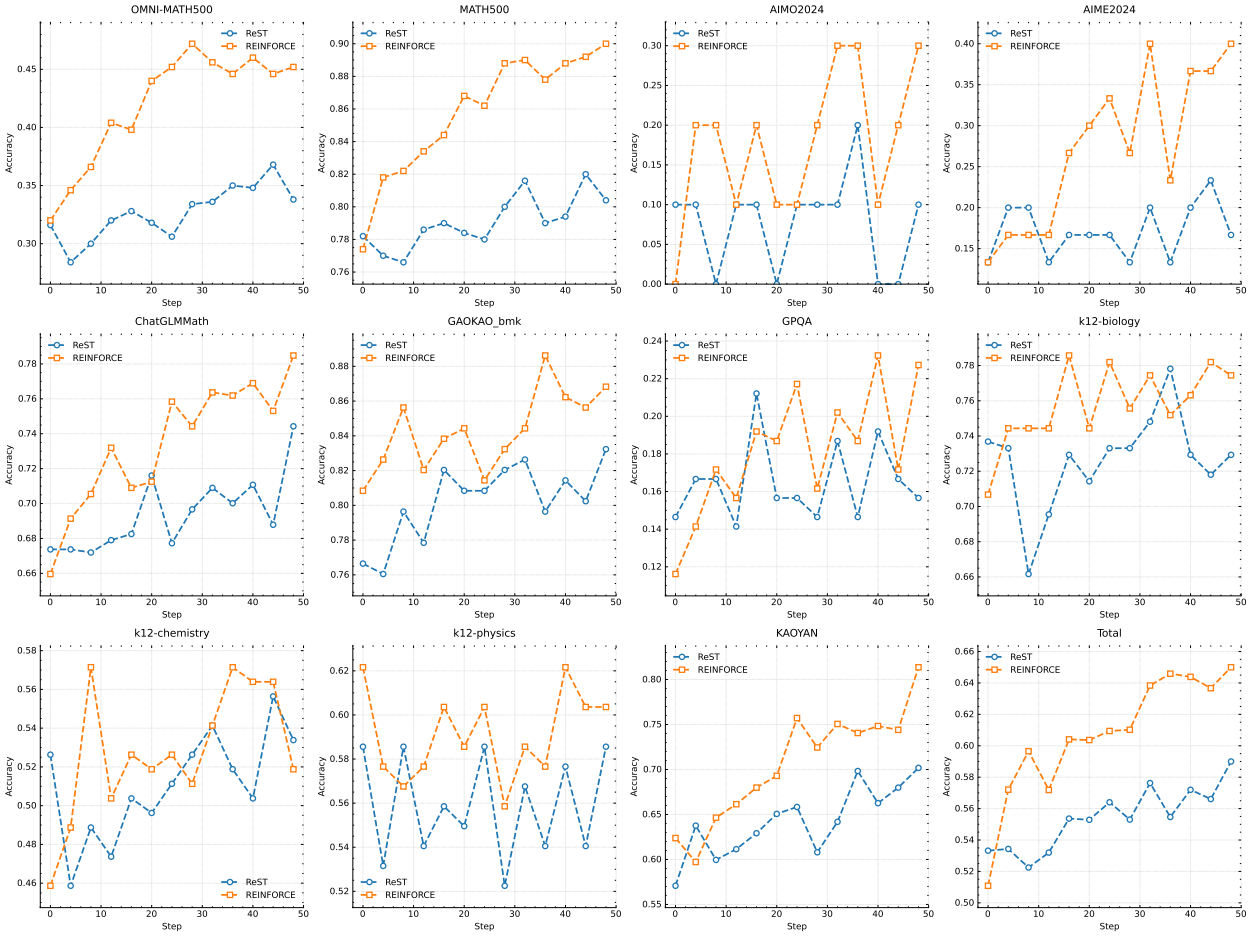


Figure 10: Comparison with using ReST for policy optimization.

efficient long-context RL training. How to further improve the efficiency and scalability of long-context RL training remains an important question moving forward.

Another contribution we made is a combination of techniques that enable improved policy optimization. Specifically, we formulate long-CoT RL with LLMs and derive a variant of online mirror descent for robust optimization. We also experiment with sampling strategies, length penalty, and optimizing the data recipe to achieve strong RL performance.

We show that strong performance can be achieved by long context scaling and improved policy optimization, even without using more complex techniques such as Monte Carlo tree search, value functions, and process reward models. In the future, it will also be intriguing to study improving credit assignments and reducing overthinking without hurting the model's exploration abilities.

We have also observed the potential of long2short methods. These methods largely improve performance of short CoT models. Moreover, it is possible to combine long2short methods with long-CoT RL in an iterative way to further increase token efficiency and extract the best performance out of a given context length budget.

References

Abbasi-Yadkori, Yasin et al. "Politex: Regret bounds for policy iteration using expert prediction". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 3692–3702.

Ahmadian, Arash et al. "Back to basics: Revisiting reinforce style optimization for learning from human feedback in llms". In: *arXiv preprint arXiv:2402.14740* (2024).

Ankner, Zachary et al. *Critique-out-Loud Reward Models*. 2024. arXiv: `2408.11791 [cs.LG]`. URL: `https://arxiv.org/abs/2408.11791`.

Berner, Christopher et al. "Dota 2 with large scale deep reinforcement learning". In: *arXiv preprint arXiv:1912.06680* (2019).

Cassano, Federico, John Gouwar, Daniel Nguyen, Sy Duy Nguyen, et al. "MultiPL-E: A Scalable and Extensible Approach to Benchmarking Neural Code Generation". In: *ArXiv* (2022). URL: https://arxiv.org/abs/2208.08227.

Cassano, Federico, John Gouwar, Daniel Nguyen, Sydney Nguyen, et al. "MultiPL-E: A Scalable and Polyglot Approach to Benchmarking Neural Code Generation". In: *IEEE Transactions on Software Engineering* 49.7 (2023), pp. 3675–3691. DOI: 10.1109/TSE.2023.3267446.

Chen, Jianlv et al. "Bge m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation". In: *arXiv preprint arXiv:2402.03216* (2024).

Chen, Xingyu et al. "Do NOT Think That Much for 2+ 3=? On the Overthinking of o1-Like LLMs". In: *arXiv preprint arXiv:2412.21187* (2024).

Everitt, Tom et al. *Reward Tampering Problems and Solutions in Reinforcement Learning: A Causal Influence Diagram Perspective*. 2021. arXiv: 1908.04734 [cs.AI]. URL: https://arxiv.org/abs/1908.04734.

Gadre, Samir Yitzhak et al. "Datacomp: In search of the next generation of multimodal datasets". In: *Advances in Neural Information Processing Systems* 36 (2024).

Grattafiori, Aaron et al. *The Llama 3 Herd of Models*. 2024. arXiv: 2407.21783 [cs.AI]. URL: https://arxiv.org/abs/2407.21783.

Gulcehre, Caglar et al. "Reinforced self-training (rest) for language modeling". In: *arXiv preprint arXiv:2308.08998* (2023).

Hendrycks, Dan et al. "Measuring Massive Multitask Language Understanding". In: *ArXiv* abs/2009.03300 (2020). URL: https://arxiv.org/abs/2009.03300.

Hoffmann, Jordan et al. *Training Compute-Optimal Large Language Models*. 2022. arXiv: 2203.15556 [cs.CL]. URL: https://arxiv.org/abs/2203.15556.

Huang, Yuzhen et al. "C-Eval: A Multi-Level Multi-Discipline Chinese Evaluation Suite for Foundation Models". In: *ArXiv* abs/2305.08322 (2023). URL: https://arxiv.org/abs/2305.08322.

Jaech, Aaron et al. "Openai o1 system card". In: *arXiv preprint arXiv:2412.16720* (2024).

Jain, Naman et al. "LiveCodeBench: Holistic and Contamination Free Evaluation of Large Language Models for Code". In: *ArXiv* abs/2403.07974 (2024). URL: https://arxiv.org/abs/2403.07974.

Joulin, Armand et al. "Bag of tricks for efficient text classification". In: *arXiv preprint arXiv:1607.01759* (2016).

Kaplan, Jared et al. *Scaling Laws for Neural Language Models*. 2020. arXiv: 2001.08361 [cs.LG]. URL: https://arxiv.org/abs/2001.08361.

Kool, Wouter, Herke van Hoof, and Max Welling. "Buy 4 reinforce samples, get a baseline for free!" In: (2019).

Kwon, Woosuk et al. "Efficient Memory Management for Large Language Model Serving with PagedAttention". In: *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*. 2023.

Laurençon, Hugo et al. "Obelics: An open web-scale filtered dataset of interleaved image-text documents". In: *Advances in Neural Information Processing Systems* 36 (2024).

Li, Jeffrey et al. "Datacomp-lm: In search of the next generation of training sets for language models". In: *arXiv preprint arXiv:2406.11794* (2024).

Li, Ming et al. "From quantity to quality: Boosting llm performance with self-guided data selection for instruction tuning". In: *arXiv preprint arXiv:2308.12032* (2023).

Li, Raymond et al. *StarCoder: may the source be with you!* 2023. arXiv: 2305.06161 [cs.CL]. URL: https://arxiv.org/abs/2305.06161.

Lightman, Hunter et al. "Let's Verify Step by Step". In: *arXiv preprint arXiv:2305.20050* (2023).

Liu, Wei et al. "What makes good data for alignment? a comprehensive study of automatic data selection in instruction tuning". In: *arXiv preprint arXiv:2312.15685* (2023).

Lozhkov, Anton et al. *StarCoder 2 and The Stack v2: The Next Generation*. 2024. arXiv: 2402.19173 [cs.SE]. URL: https://arxiv.org/abs/2402.19173.

Lu, Pan et al. "Mathvista: Evaluating mathematical reasoning of foundation models in visual contexts". In: *arXiv preprint arXiv:2310.02255* (2023).

McAleese, Nat et al. *LLM Critics Help Catch LLM Bugs*. 2024. arXiv: 2407.00215 [cs.SE]. URL: https://arxiv.org/abs/2407.00215.

Mei, Jincheng et al. "On principled entropy exploration in policy optimization". In: *Proceedings of the 28th International Joint Conference on Artificial Intelligence*. 2019, pp. 3130–3136.

Muennighoff, Niklas et al. *Scaling Data-Constrained Language Models*. 2023. arXiv: 2305.16264 [cs.CL]. URL: https://arxiv.org/abs/2305.16264.

Nachum, Ofir et al. "Bridging the gap between value and policy based reinforcement learning". In: *Advances in neural information processing systems* 30 (2017).

OpenAI. "Learning to reason with LLMs". In: (2024). URL: https://openai.com/index/learning-to-reason-with-llms/.
Ouyang, Long et al. "Training language models to follow instructions with human feedback". In: *Advances in neural information processing systems* 35 (2022), pp. 27730–27744.
Pan, Alexander, Kush Bhatia, and Jacob Steinhardt. "The Effects of Reward Misspecification: Mapping and Mitigating Misaligned Models". In: *International Conference on Learning Representations*. 2022. URL: https://openreview.net/forum?id=JYtwGwIL7ye.
Paster, Keiran et al. "Openwebmath: An open dataset of high-quality mathematical web text". In: *arXiv preprint arXiv:2310.06786* (2023).
Penedo, Guilherme et al. "The fineweb datasets: Decanting the web for the finest text data at scale". In: *arXiv preprint arXiv:2406.17557* (2024).
Qin, Ruoyu et al. *Mooncake: A KVCache-centric Disaggregated Architecture for LLM Serving*. 2024. arXiv: 2407.00079 [cs.DC]. URL: https://arxiv.org/abs/2407.00079.
Rafailov, Rafael et al. "Direct preference optimization: Your language model is secretly a reward model". In: *Advances in Neural Information Processing Systems* 36 (2024).
Schuhmann, Christoph et al. "Laion-5b: An open large-scale dataset for training next generation image-text models". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 25278–25294.
Shoeybi, Mohammad et al. *Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism*. 2020. arXiv: 1909.08053 [cs.CL]. URL: https://arxiv.org/abs/1909.08053.
Silver, David et al. "Mastering the game of go without human knowledge". In: *nature* 550.7676 (2017), pp. 354–359.
Snell, Charlie et al. "Scaling llm test-time compute optimally can be more effective than scaling model parameters". In: *arXiv preprint arXiv:2408.03314* (2024).
Su, Dan et al. "Nemotron-CC: Transforming Common Crawl into a Refined Long-Horizon Pretraining Dataset". In: *arXiv preprint arXiv:2412.02595* (2024).
Su, Jianlin et al. "Roformer: Enhanced transformer with rotary position embedding". In: *Neurocomputing* 568 (2024), p. 127063.
Team, Gemini et al. *Gemini: A Family of Highly Capable Multimodal Models*. 2024. arXiv: 2312.11805 [cs.CL]. URL: https://arxiv.org/abs/2312.11805.
Tomar, Manan et al. "Mirror descent policy optimization". In: *arXiv preprint arXiv:2005.09814* (2020).
Vaswani, Ashish et al. "Attention is All you Need". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.
Villalobos, Pablo et al. *Will we run out of data? Limits of LLM scaling based on human-generated data*. 2024. arXiv: 2211.04325 [cs.LG]. URL: https://arxiv.org/abs/2211.04325.
Vinyals, Oriol et al. "Grandmaster level in StarCraft II using multi-agent reinforcement learning". In: *nature* 575.7782 (2019), pp. 350–354.
Wang, Ke et al. "Measuring multimodal mathematical reasoning with math-vision dataset". In: *arXiv preprint arXiv:2402.14804* (2024).
Wei, Haoran et al. "General OCR Theory: Towards OCR-2.0 via a Unified End-to-end Model". In: *arXiv preprint arXiv:2409.01704* (2024).
Wei, Jason et al. "Chain-of-thought prompting elicits reasoning in large language models". In: *Advances in neural information processing systems* 35 (2022), pp. 24824–24837.
Wu, Yangzhen et al. "Inference scaling laws: An empirical analysis of compute-optimal inference for problem-solving with language models". In: *arXiv preprint arXiv:2408.00724* (2024).
Xu, Liang et al. "CLUE: A Chinese Language Understanding Evaluation Benchmark". In: *International Conference on Computational Linguistics*. 2020. URL: https://arxiv.org/abs/2004.05986.
Yang, Enneng et al. "Model merging in llms, mllms, and beyond: Methods, theories, applications and opportunities". In: *arXiv preprint arXiv:2408.07666* (2024).
Yao, Shunyu et al. "Tree of thoughts: Deliberate problem solving with large language models". In: *Advances in Neural Information Processing Systems* 36 (2024).
Yue, Xiang, Yuansheng Ni, et al. "Mmmu: A massive multi-discipline multimodal understanding and reasoning benchmark for expert agi". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2024, pp. 9556–9567.
Yue, Xiang, Xingwei Qu, et al. "Mammoth: Building math generalist models through hybrid instruction tuning". In: *arXiv preprint arXiv:2309.05653* (2023).
Zhang, Lunjun et al. "Generative verifiers: Reward modeling as next-token prediction, 2024". In: *URL https://arxiv.org/abs/2408.15240* (2024).
Zheng, Lianmin et al. *SGLang: Efficient Execution of Structured Language Model Programs*. 2024. arXiv: 2312.07104 [cs.AI]. URL: https://arxiv.org/abs/2312.07104.

Zhou, Jeffrey et al. “Instruction-Following Evaluation for Large Language Models”. In: *ArXiv* abs/2311.07911 (2023). URL: https://arxiv.org/abs/2311.07911.

Zhu, Wanrong et al. “Multimodal c4: An open, billion-scale corpus of images interleaved with text”. In: *Advances in Neural Information Processing Systems* 36 (2024).

Appendix

A Contributions

Research & Development

Angang Du
Bofei Gao
Changjiu Jiang
Cheng Chen
Cheng Li
Chenjun Xiao
Chenzhuang Du
Dehao Zhang
Enming Yuan
Enzhe Lu
Flood Sung
Guokun Lai
Haiqing Guo
Han Zhu
Hao Ding
Hao Hu
Hao Yang
Hao Zhang
Haotian Yao
Haotian Zhao
Haoyu Lu
Hongcheng Gao
Huan Yuan
Huabin Zheng
Jingyuan Liu
Jianlin Su
Jianzhou Wang
Jin Zhang
Junjie Yan
Lidong Shi
Longhui Yu
Mengnan Dong
Neo Zhang
Qiwei Pan
Qucheng Gong
Shupeng Wei
Shaowei Liu
Tao Jiang
Weimin Xiong
Weiran He
Weihao Gao
Weixiao Huang
Wenhao Wu
Wenyang He
Xianqing Jia
Xinran Xu
Xinyu Zhou
Xinxing Zu
Yang Li
Yangyang Hu
Yangyang Liu
Yanru Chen
Yejie Wang
Yidao Qin
Yibo Liu
Yiping Bao
Yulun Du
Yuzhi Wang
Yuxin Wu
Y. Charles
Zaida Zhou
Zhaoji Wang
Zhaowei Li
Zheng Zhang
Zhexu Wang
Zhiqi Huang
Zhilin Yang
Ziyao Xu
Zonghan Yang

Data Annotation

Chuning Tang
Congcong Wang
Fengxiang Tang
Guangda Wei
Haoze Li
Haozhen Yu
Jia Chen
Jianhang Guo
Jie Zhao
Junyan Wu
Ling Ye
Shengling Ma
Sihan Cao
Siying Huang
Xianghui Wei
Ying Yang
Yangyang Liu
Zhen Zhu
Zihao Huang

The listing of authors is in alphabetical order based on their first names.

B Qualitative Examples

Due to the space limit, please refer to our github homepage [3].

C Pretraining

Reinforcement learning (RL) efficiency is closely tied to the performance of the underlying base model. Frontier models such as Gemini(Team et al. 2024) and Llama(Grattafiori et al. 2024) highlight the importance of pretraining data quality in achieving high performance. However, many recent open-source models lack full transparency regarding their data processing pipelines and recipes, creating challenges for broader community understanding. While we are not open-sourcing our proprietary model at this time, we are committed to providing a comprehensive disclosure of our data pipeline and methodologies. In this section, we focus primarily on the multimodal pretraining data recipe, followed by a brief discussion of the model architecture and training stages.

C.1 Language Data

Our pretrain corpus is designed to provide comprehensive and high-quality data for training large language models (LLMs). It encompasses five domains: English, Chinese, Code, Mathematics & Reasoning, and Knowledge. We employ sophisticated filtering and quality control mechanisms for each domain to ensure the highest quality training data. For all pretrain data, we conducted rigorous individual validation for each data source to assess its specific contribution to the overall training recipe. This systematic evaluation ensures the quality and effectiveness of our diverse data composition.

English and Chinese textual data we developed a multi-dimensional quality filtering framework that combines multiple scoring methods to reduce individual biases and ensure comprehensive quality assessment. Our framework incorporates:

1. **Rule-based filtering**: We implement domain-specific heuristics to remove problematic content, including duplicate content, machine-translated text, and low-quality web scrapes. We also filter out documents with excessive special characters, unusual formatting, or spam patterns.
2. **FastText-based classification**: We trained specialized FastText(Joulin et al. 2016; J. Li et al. 2024) models to identify content quality based on linguistic features and semantic coherence. This helps identify documents with natural language flow and proper grammatical structure.
3. **Embedding-based similarity analysis**: Using document embeddings (Jianlv Chen et al. 2024), we compute document-level similarity scores to identify and remove near-duplicates while preserving semantically valuable variations. This approach helps maintain diversity in our training corpus.
4. **LLM-based quality assessment**: Following (Penedo et al. 2024), we leverage LLMs to score documents based on coherence, informativeness, and potential educational value. This method is particularly effective at identifying nuanced quality indicators that simpler methods might miss.

The final quality score for each document is calculated as a combination of these individual scores. Based on extensive empirical analysis, we implement dynamic sampling rates, where high-quality documents are upsampled, while low-quality documents are downsampled during training.

Code data The code data primarily consists of two categories. For the pure code data derived from code files, we adhered to the methodology of BigCode (R. Li et al. 2023; Lozhkov et al. 2024) and conducted a comprehensive preprocessing of the dataset. Initially, we eliminated miscellaneous languages and applied a rule-based cleaning procedure to enhance data quality. Subsequently, we addressed language imbalance through strategic sampling techniques. Specifically, markup languages such as JSON, YAML, and YACC were down-sampled, while 32 major programming languages, including Python, C, C++, Java, and Go, were up-sampled to ensure a balanced representation. Regarding the text-code interleaved data sourced from various data sources, we use an embedding-based method to recall high-quality data. This approach ensures the diversity of the data and maintains its high quality.

Math & Reasoning data The mathematics and reasoning component of our dataset is crucial for developing strong analytical and problem-solving capabilities. The mathematical pre-training data are mainly retrieved from web text

[3] https://github.com/MoonshotAI/Kimi-k1.5

and PDF documents collected from publicly available internet sources. (Paster et al. 2023) Initially, we discovered that our general-domain text extraction, data cleaning process and OCR models exhibited high false negative rates in the mathematical domain. Therefore, we first developed specialized data cleaning procedures and OCR models specifically for mathematical content, aiming to maximize the recall rate of mathematical data. Subsequently, we implemented a two-stage data cleaning process:

1. Using FastText model for initial cleaning to remove most irrelevant data.
2. Utilizing a fine-tuned language model to further clean the remaining data, resulting in high-quality mathematical data.

Knowledge data The knowledge corpus is meticulously curated to ensure a comprehensive coverage in academic disciplines. Our knowledge base primarily consists of academic exercises, textbooks, research papers, and other general educational literature. A significant portion of these materials is digitized through OCR processing, for which we have developed proprietary models optimized for academic content, particularly for handling mathematical formulas and special symbols.

We employ internal language models to annotate documents with multi-dimensional labels, including:

1. OCR quality metrics to assess recognition accuracy
2. Educational value indicators measuring pedagogical relevance
3. Document type classification (e.g., exercises, theoretical materials)

Based on these multi-dimensional annotations, we implement a sophisticated filtering and sampling pipeline. First and foremost, documents are filtered through OCR quality thresholds. Our OCR quality assessment framework places special attention on detecting and filtering out common OCR artifacts, particularly repetitive text patterns that often indicate recognition failures.

Beyond basic quality control, we carefully evaluate the educational value of each document through our scoring system. Documents with high pedagogical relevance and knowledge depth are prioritized, while maintaining a balance between theoretical depth and instructional clarity. This helps ensure that our training corpus contains high-quality educational content that can effectively contribute to the model's knowledge acquisition.

Finally, to optimize the overall composition of our training corpus, the sampling strategy for different document types is empirically determined through extensive experimentation. We conduct isolated evaluations to identify document subsets that contribute most significantly to the model's knowledge acquisition capabilities. These high-value subsets are upsampled in the final training corpus. However, to maintain data diversity and ensure model generalization, we carefully preserve a balanced representation of other document types at appropriate ratios. This data-driven approach helps us optimize the trade-off between focused knowledge acquisition and broad generalization capabilities.

C.2 Multimodal Data

Our multi-modal pretraining corpus is designed to provide high-quality data that enables models to process and understand information from multiple modalities, including text, images, and videos. To this end, we also have curated high-quality data from five categories—captioning, interleaving, OCR (Optical Character Recognition), knowledge, and general question answering—to form the corpus.

When constructing our training corpus, we developed several multi-modal data processing pipelines to ensure data quality, encompassing filtering, synthesis, and deduplication. Establishing an effective multi-modal data strategy is crucial during the joint training of vision and language, as it both preserves the capabilities of the language model and facilitates alignment of knowledge across diverse modalities.

We provide a detailed description of these sources in this section, which is organized into the following categories:

Caption data Our caption data provides the model with fundamental modality alignment and a broad range of world knowledge. By incorporating caption data, the multi-modal LLM gains wider world knowledge with high learning efficiency. We have integrated various open-source Chinese and English caption datasets like (Schuhmann et al. 2022; S. Y. Gadre et al. 2024) and also collected substantial in-house caption data from multiple sources. However, throughout the training process, we strictly limit the proportion of synthetic caption data to mitigate the risk of hallucination stemming from insufficient real-world knowledge.

For general caption data, we follow a rigorous quality control pipeline that avoids duplication and maintain high image-text correlation. We also vary image resolution during pretraining to ensure that the vision tower remains effective when processing images of both high- and low-resolution.

Image-text interleaving data During the pretraining phase, model is benefit from interleaving data for many aspects, for example, multi-image comprehension ability can be boosted by interleaving data; interleaving data always provide detailed knowledge for the given image; a longer multi-modal context learning ability can also be gained by the interleaving data. What's more, we also find that interleaving data can contributes positively to maintaining the model's language abilities. Thus, image-text interleaving data is an important part in our training corpus. Our multi-modal corpus considered open-sourced interleave datasets like (Zhu et al. 2024; Laurençon et al. 2024) and also constructed large-scale in-house data using resources like textbooks, webpages and tutorials. Further, we also find that synthesizing the interleaving data benefits the performance of multi-modal LLM for keeping the text knowledges. To ensure each image's knowledge is sufficiently studied, for all the interleaving data, other than the standard filtering, deduping and other quality control pipeline, we also integrated a data reordering procedure for keeping all the image and text in the correct order.

OCR data Optical Character Recognition (OCR) is a widely adopted technique that converts text from images into an editable format. In k1.5, a robust OCR capability is deemed essential for better aligning the model with human values. Accordingly, our OCR data sources are diverse, ranging from open-source to in-house datasets, and encompassing both clean and augmented images.

In addition to the publicly available data, we have developed a substantial volume of in-house OCR datasets, covering multilingual text, dense text layouts, web-based content, and handwritten samples. Furthermore, following the principles outlined in OCR 2.0 (H. Wei et al. 2024), our model is also equipped to handle a variety of optical image types, including figures, tables, geometry diagrams, mermaid plots, and natural scene text. We apply extensive data augmentation techniques—such as rotation, distortion, color adjustments, and noise addition—to enhance the model's robustness. As a result, our model achieves a high level of proficiency in OCR tasks.

Knowledge data The concept of multi-modal knowledge data is analogous to the previously mentioned text pretraining data, except here we focus on assembling a comprehensive repository of human knowledge from diverse sources to further enhance the model's capabilities. For example, carefully curated geometry data in our dataset is vital for developing visual reasoning skills, ensuring the model can interpret the abstract diagrams created by humans.

Our knowledge corpus adheres to a standardized taxonomy to balance content across various categories, ensuring diversity in data sources. Similar to text-only corpora, which gather knowledge from textbooks, research papers, and other academic materials, multi-modal knowledge data employs both a layout parser and an OCR model to process content from these sources. While we also include filtered data from internet-based and other external resources.

Because a significant portion of our knowledge corpus is sourced from internet-based materials, infographics can cause the model to focus solely on OCR-based information. In such cases, relying exclusively on a basic OCR pipeline may limit training effectiveness. To address this, we have developed an additional pipeline that better captures the purely textual information embedded within images.

General QA Data During the training process, we observed that incorporating a substantial volume of high-quality QA datasets into pretraining offers significant benefits. Specifically, we included rigorous academic datasets addressing tasks such as grounding, table/chart question answering, web agents, and general QA. In addition, we compiled a large amount of in-house QA data to further enhance the model's capabilities. To maintain balanced difficulty and diversity, we applied scoring models and meticulous manual categorization to our general question answering dataset, resulting in overall performance improvements.

C.3 Model Architecture

Kimi k-series models employ a variant of the Transformer decoder (Vaswani et al. 2017) that integrates multimodal capabilities alongside improvements in architecture and optimization strategies, illustrated in Figure 11. These advancements collectively support stable large-scale training and efficient inference, tailored specifically to large-scale reinforcement learning and the operational requirements of Kimi users.

Extensive scaling experiments indicate that most of the base model performance comes from improvements in the quality and diversity of the pretraining data. Specific details regarding model architecture scaling experiments lie beyond the scope of this report and will be addressed in future publications.

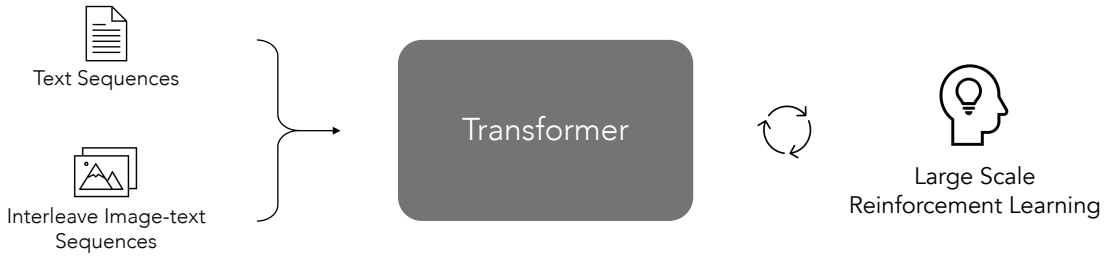


Figure 11: Kimi k1.5 supports interleaved images and text as input, leveraging large-scale reinforcement learning to enhance the model's reasoning capabilities.

C.4 Training Stages

The Kimi k1.5 model is trained in three stages: the vision-language pretraining stage, the vision-language cooldown stage, and the long-context activation stage. Each stage of the Kimi k1.5 model's training focuses on a particular capability enhancement.

Vision-language pretraining stage In this stage, the model is firstly trained solely on language data, establishing a robust language model foundation. Then the model is gradually introduced to interleaved vision-language data, acquiring multimodal capabilities. The visual tower is initially trained in isolation without updating the language model parameters, then we unfreeze the language model layers, and ultimately increase the proportion of vision-text data to 30%. The final data mixtures and their respective weights were determined through ablation studies conducted on smaller models.

Vision-language cooldown stage The second stage serves as a cooldown phase, where the model is continue trained with high-quality language and vision-language datasets to ensure superior performance. Through empirical investigation, we observed that the incorporation of synthetic data during the cooldown phase yields significant performance improvements, particularly in mathematical reasoning, knowledge-based tasks, and code generation. The English and Chinese components of the cooldown dataset are curated from high-fidelity subsets of the pre-training corpus. For math, knowledge, and code domains, we employ a hybrid approach: utilizing selected pre-training subsets while augmenting them with synthetically generated content. Specifically, we leverage existing mathematical, knowledge and code corpora as source material to generate question-answer pairs through a proprietary language model, implementing rejection sampling techniques to maintain quality standards (Yue, Qu, et al. 2023; D. Su et al. 2024). These synthesized QA pairs undergo comprehensive validation before being integrated into the cooldown dataset.

Long-context activation stage Finally, in the third stage, k1.5 is trained with upsampled long-context cooldown data, enabling it to process extended sequences and support tasks that demand longer context. To ensure excellent long-text capabilities of the base model, we upsampled long context data and used 40% full attention data and 60% partial attention data during long context training. The full attention data came partly from high-quality natural data and partly from synthetic long context Q&A and summary data. The partial attention data came from uniform sampling of cooldown data. The RoPE frequency (J. Su et al. 2024) was set to 1,000,000. During this stage, we gradually extended length activation training by increasing the maximum sequence length from 4,096 to 32,768, and ultimately to 131,072.

D Evaluation Details

Text Benchmark MMLU (Hendrycks et al. 2020) covers 57 subjects in STEM, the humanities, social sciences, and more. It ranges in difficulty from an elementary level to an advanced professional level, and it tests both world knowledge and problem-solving ability.

IF-Eval (J. Zhou et al. 2023) is a benchmark for evaluating large language models' ability to follow verifiable instructions. There are 500+ prompts with instructions such as "write an article with more than 800 words", etc. Due to a version shift, the number of IFEval reported in Table 3 derived from an intermediate model. We will update the scores based on the final model.

CLUEWSC (L. Xu et al. 2020) is a coreference resolution task in CLUE benchmark, requiring models to determine if a pronoun and a noun phrase in a sentence co-refer, with data from Chinese fiction books.

C-EVAL (Y. Huang et al. 2023) is a comprehensive Chinese evaluation suite for assessing advanced knowledge and reasoning abilities of foundation models. It includes 13,948 multiple-choice questions across 52 disciplines and four difficulty levels.

Reasoning Benchmark HumanEval-Mul is a subset of Multipl-E (Cassano, Gouwar, D. Nguyen, S. D. Nguyen, et al. 2022). MultiPL-E extends the HumanEval benchmark and MBPP benchmark to 18 languages that encompass a range of programming paradigms and popularity. We choose HumanEval translations in 8 mainstream programming languages (Python, Java, Cpp, C#, JavaScript, TypeScript, PHP, and Bash).

LiveCodeBench (Jain et al. 2024) serves as a comprehensive and contamination-free benchmark for assessing large language models (LLMs) in coding tasks. It features live updates to prevent data contamination, holistic evaluation across multiple coding scenarios, high-quality problems and tests, and balanced problem difficulty. We test short-CoT model with questions from 2408-2411 (release v4), and long-CoT model with questions from 2412-2502 (release v5).

AIME 2024 comprises the competition questions for the AIME in 2024. The AIME is a prestigious, invitation-only math contest for top high school students, assessing advanced math skills and requiring solid foundation and high logical thinking.

MATH-500 (Lightman et al. 2023) is a comprehensive mathematics benchmark that contains 500 problems on various mathematics topics including algebra, calculus, probability, and more. Tests both computational ability and mathematical reasoning. Higher scores indicate stronger mathematical problem-solving capabilities.

Codeforces is a well-known online judge platform and serves as a popular testbed for evaluating long-CoT coding models. To achieve higher rankings in the Div2 and Div3 competitions, we utilize majority voting on the code snippets generated by the k1.5 long-CoT model, employing test cases that are also generated by the same model.

Image Benchmark MMMU (Yue, Ni, et al. 2024) encompasses a carefully curated collection of 11.5K multimodal questions sourced from college exams, quizzes, and textbooks. These questions span six major academic fields: Art & Design, Business, Science, Health & Medicine, Humanities & Social Science, and Tech & Engineering.

The MATH-Vision (MATH-V) (K. Wang et al. 2024) dataset is a carefully curated collection of 3,040 high-quality mathematical problems with visual contexts that are sourced from real math competitions. It covers 16 distinct mathematical disciplines and is graded across 5 levels of difficulty. This dataset offers a comprehensive and diverse set of challenges, making it ideal for evaluating the mathematical reasoning abilities of LMMs.

MathVista (Lu et al. 2023) is a benchmark that integrates challenges from a variety of mathematical and visual tasks, demanding participants to exhibit fine-grained, deep visual understanding along with compositional reasoning to successfully complete the tasks.