

本文由 AINLP 公众号整理翻译，更多 LLM 资源请扫码关注!

**AINLP**

我爱自然语言处理

一个有趣有AI的自然语言处理社区



长按扫码关注我们

---

# Magistral

---



## Abstract

We introduce Magistral, Mistral’s first reasoning model and our own scalable reinforcement learning (RL) pipeline. Instead of relying on existing implementations and RL traces distilled from prior models, we follow a ground up approach, relying solely on our own models and infrastructure. Notably, we demonstrate a stack that enabled us to explore the limits of pure RL training of LLMs, present a simple method to force the reasoning language of the model, and show that RL on text data alone maintains most of the initial checkpoint’s capabilities. We find that RL on text maintains or improves multimodal understanding, instruction following and function calling. We present Magistral Medium, trained for reasoning on top of Mistral Medium 3 with RL alone, and we open-source Magistral Small (Apache 2.0) which further includes cold-start data from Magistral Medium.

## 1 Introduction

Enhancing the reasoning abilities of large language models (LLMs) has emerged as a key frontier in modern AI research. Reasoning models such as o1 [Jaech et al., 2024] differ widely from classic chatbots, leveraging longer chains-of-thought to improve performance on complex tasks. The seminal work by DeepSeek-AI et al. [2025] gave the community crucial insights on the Reinforcement Learning from Verifiable Rewards (RLVR) recipe, for creating reasoning models at scale.

In this paper, we introduce Mistral’s first reasoning models: Magistral Small and Magistral Medium, based on the Mistral Small 3 and Mistral Medium 3 models respectively, and outline our proposed RLVR framework in detail. The key contributions of our paper are the following:

- We present in detail how we trained Magistral Medium with RL alone, with no distillation from pre-existing reasoning models, yielding a nearly 50% boost in AIME-24 (pass@1).
- We discuss in depth the infrastructure and design choices that enable large-scale online RL. Our asynchronous system enables fast, continuous RL training by updating generators frequently without interrupting them, balancing efficiency with on-policy-ness.
- We present a simple yet effective strategy to make the model multilingual, where both the chain-of-thought and the final response are written in the user’s language.
- We contribute insights that add to, or contradict, existing RLVR literature, for example on whether RL can improve upon the distillation SFT baseline for small models. We also show that multimodal reasoning capabilities emerge with online RL with textual data on top of a multimodal model. We share the results of our unsuccessful experiments.
- We release the weights of Magistral Small (24B) under the Apache 2 license<sup>1</sup>.

---

<sup>1</sup><https://huggingface.co/mistralai/Magistral-Small-2506>

---

# Magistral

---



## 摘要

我们介绍Magistral，Mistral的第一个推理模型以及我们自主开发的可扩展强化学习（RL）管道。我们没有依赖现有的实现和从先前模型中提取的RL轨迹，而是采用自底向上的方法，完全依靠我们自己的模型和基础设施。值得注意的是，我们展示了一个堆栈，使我们能够探索纯RL训练LLMs的极限，提出了一种简单的方法来强制模型的推理语言，并显示仅在文本数据上进行RL可以保持大部分初始检查点的能力。我们发现，在文本上的RL可以保持或提升多模态理解、指令执行和函数调用能力。我们还推出了Magistral Medium，该模型在仅通过RL的基础上，在Mistral Medium 3之上进行推理训练，并开源了Magistral Small（Apache 2.0），该模型还包括来自Magistral Medium的冷启动数据。

## 1 引言

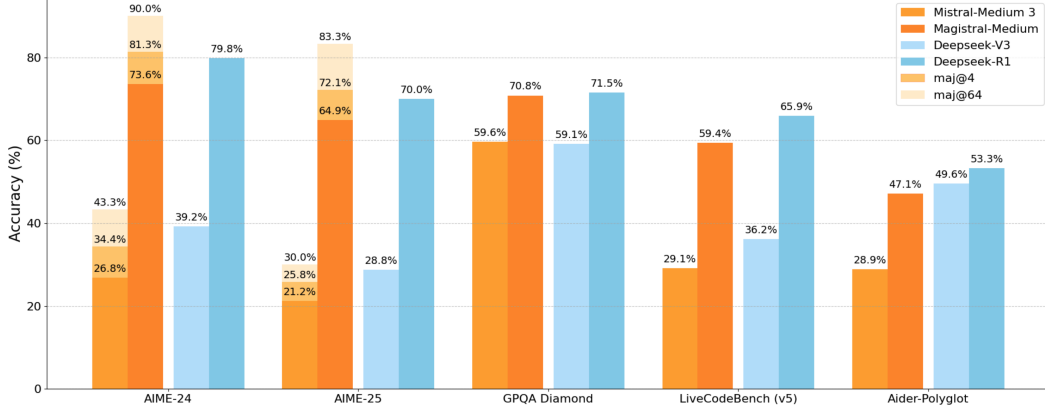
提升大型语言模型（LLMs）的推理能力已成为现代人工智能研究的一个关键前沿。推理模型如 o1 [Jaech et al., 2024] 与经典的聊天机器人有很大不同，它们利用更长的思维链来提升在复杂任务中的表现。DeepSeek-AI 等人 [2025] 的开创性工作为社区提供了关于可验证奖励强化学习（RLVR）方案的关键见解，用于大规模构建推理模型。

在本文中，我们介绍了Mistral的第一个推理模型：Magistral Small和Magistral Medium，分别基于Mistral Small 3和Mistral Medium 3模型，并详细阐述了我们提出的RLVR框架。本文的主要贡献如下：

- 我们详细介绍了如何仅使用强化学习（RL）训练Magistral Medium，没有借助现有推理模型的蒸馏，从而在AIME-24（pass@1）中实现了近50%的提升。
- 我们深入讨论了支持大规模在线强化学习的基础设施和设计选择。我们的异步系统通过频繁更新生成器而不中断它们，实现了快速、连续的强化学习训练，在效率和策略一致性之间取得了平衡。
- 我们提出了一种简单而有效的策略，使模型具备多语言能力，其中链式思维和最终回答都用用户的语言书写。
- 我们提供的见解补充或与现有的RLVR文献相矛盾，例如关于RL是否能在小模型的蒸馏SFT基线之上取得改进。我们还展示了在多模态模型基础上，通过在线RL结合文本数据，多模态推理能力得以出现。我们分享了我们未成功的实验结果。
- 我们在Apache 2许可证下发布Magistral Small（24B）的权重<sup>1</sup>。

---

<sup>1</sup><https://huggingface.co/mistralai/Magistral-Small-2506>



**Figure 1: Performance of Magistral Medium on common reasoning benchmarks.** We highlight the strength of our proposed RLVR framework, which yields a 50% increase in AIME-24 (pass@1) over the initial Mistral Medium 3 checkpoint, *without any cold-start reasoning traces*. We compare against analogous results from [DeepSeek-AI et al., 2025], which show RL improvements from DeepSeek-v3 to DeepSeek-R1 (January 25). Magistral Medium reaches 90% accuracy on AIME-24 with majority voting.

The paper is organized as follows: Section 2 details the RL algorithm we used, along with the design choices implemented to guide the reasoning models in terms of language and format; Section 3 presents our scalable infrastructure that supports efficient training on a large cluster of GPUs; Section 4 discusses the data selection process we employed for efficient and effective training; Section 5 presents the performance of Magistral on reasoning and multilingual benchmarks; Section 6 shows the ablations done to motivate the training choices; Section 7 presents a PCA-based study of the model weights’ trajectory during RL, demonstrates that RL on text data preserves or even improves multimodal capabilities, and includes methods that worked poorly for Magistral; Section 8 shows that one can train a model to perform on par with R1 with distillation followed by RL, which we did not use for Magistral Medium; Finally, we conclude with some future directions in Section 9.

## 2 Methodology

In this section, we outline the training methodology used to develop the Magistral models. This includes our optimizations of the GRPO algorithm for training stability (Section 2.1) and our training reward to improve both mathematical and coding capabilities, while ensuring the model adheres to proper format, length, and language usage (Section 2.2).

### 2.1 Reinforcement learning algorithm

We use Group Relative Policy Optimization (GRPO) [Shao et al., 2024] as our RL algorithm. Unlike PPO [Schulman et al., 2017], GRPO eliminates the need for a ‘critic model’, and instead uses the average reward from multiple generations per prompt from the policy to compute a baseline for advantage calculation. Specifically, GRPO optimizes the policy  $\pi_\theta$  to maximize the following objective:

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}_{q \sim P(Q), \{o_i\}_{i=1}^G \sim \pi_{\theta_{\text{old}}}(\cdot|q)} \left[ \sum_{i=1}^G \sum_{t=1}^{|o_i|} \frac{1}{|o_i|} \left( \min \left[ \frac{\pi_\theta(o_{i,t}|q, o_{i,<t})}{\pi_{\theta_{\text{old}}}(o_{i,t}|q, o_{i,<t})} \hat{A}_{i,t}, \text{clip}\left(\frac{\pi_\theta(o_{i,t}|q, o_{i,<t})}{\pi_{\theta_{\text{old}}}(o_{i,t}|q, o_{i,<t})}, 1 - \varepsilon, 1 + \varepsilon\right) \hat{A}_{i,t} \right] - \beta D_{\text{KL}}[\pi_\theta(\cdot|q) \parallel \pi_{\text{ref}}(\cdot|q)] \right) \right],$$

where  $q$  represents queries drawn from the input dataset,  $o$  represents the generation of the model,  $\varepsilon$  is the PPO clipping threshold,  $\beta$  is the KL penalty coefficient, and  $D_{\text{KL}}$  denotes the Kullback–Leibler divergence between the current policy  $\pi_\theta$  and the reference policy  $\pi_{\text{ref}}$ . The relative advantage, or the group normalized advantage, is given by  $\hat{A}_{i,t} = \frac{r_{i,t} - \mu}{\sigma}$  where  $\mu$  and  $\sigma$  are the mean and standard

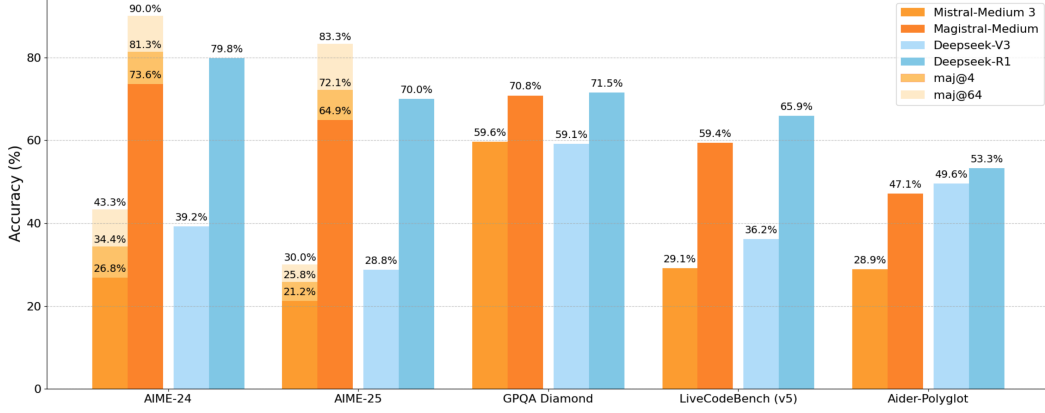


图1: Magistral Medium在常见推理基准测试中的表现。我们强调了所提出的RLVR框架的优势，该框架在AIME-24 (pass@1) 上比初始的Mistral Medium 3检查点*without any cold-start reasoning traces*提高了50%。我们与[DeepSeek-AI等, 2025]的类似结果进行了比较，显示从DeepSeek-v3到DeepSeek-R1 (2025年1月)，RL性能有所提升。Magistral Medium通过多数投票在AIME-24上达到了90%的准确率。

本文结构如下：第2节详细介绍了我们使用的强化学习（RL）算法，以及为引导推理模型在语言和格式方面的设计选择；第3节展示了支持在大型GPU集群上高效训练的可扩展基础设施；第4节讨论了我們采用的数据选择过程，以实现高效且有效的训练；第5节介绍了Magistral在推理和多语言基准测试中的性能；第6节展示了为激发训练选择而进行的消融实验；第7节基于主成分分析（PCA）研究了RL过程中模型权重的轨迹，证明在文本数据上的RL可以保持甚至提升多模态能力，并包括一些在Magistral中效果不佳的方法；第8节显示通过蒸馏后再进行RL，可以训练出与R1性能相当的模型，而我们未在Magistral Medium中使用此方法；最后，第9节展望了一些未来的研究方向。

## 2 方法论

在本节中，我们概述了用于开发Magistral模型的训练方法。这包括我们对GRPO算法的优化，以提高训练的稳定性（第2.1节）以及我们的训练奖励，以提升数学和编码能力，同时确保模型遵循正确的格式、长度和语言使用（第2.2节）。

### 2.1 强化学习算法

我们使用群体相对策略优化（GRPO）[Shao 等, 2024] 作为我们的强化学习算法。与 PPO [Schulman 等, 2017] 不同，GRPO 省去了“评论模型”的需求，而是使用来自策略的每个提示的多代平均奖励来计算优势基线。具体而言，GRPO 优化策略  $\pi_\theta$  以最大化以下目标：

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}_{q \sim P(Q), \{o_i\}_{i=1}^G \sim \pi_{\theta_{\text{old}}}(\cdot|q)} \left[ \sum_{i=1}^G \sum_{t=1}^{|o_i|} \frac{1}{|o_i|} \left( \min \left[ \frac{\pi_\theta(o_{i,t}|q, o_{i,<t})}{\pi_{\theta_{\text{old}}}(o_{i,t}|q, o_{i,<t})} \hat{A}_{i,t}, \text{clip}\left(\frac{\pi_\theta(o_{i,t}|q, o_{i,<t})}{\pi_{\theta_{\text{old}}}(o_{i,t}|q, o_{i,<t})}, 1 - \varepsilon, 1 + \varepsilon\right) \hat{A}_{i,t} \right] - \beta D_{\text{KL}}[\pi_\theta(\cdot|q) \parallel \pi_{\text{ref}}(\cdot|q)] \right) \right],$$

其中  $q$  代表从输入数据集中抽取的查询， $o$  代表模型的生成， $\varepsilon$  是 PPO 剪裁阈值， $\beta$  是 KL 惩罚系数，而  $D_{\text{KL}}$  表示当前策略  $\pi_\theta$  与参考策略  $\pi_{\text{ref}}$  之间的 Kullback-Leibler 散度。相对优势，或称为组归一化优势，由  $\hat{A}_{i,t} = \frac{r_{i,t} - \mu}{\sigma}$  给出，其中  $\mu$  和  $\sigma$  分别是均值和标准差。

deviation of rewards computed within a single group. Building on prior work adapting GRPO for reasoning tasks [Yu et al., 2025, Liu et al., 2025, Hu et al., 2025], we introduced several modifications:

**Eliminating KL divergence.** The KL divergence penalty constrains the online policy from deviating too much from a reference policy, helping to maintain alignment with the initial model. However, in GRPO, the policy diverges substantially regardless, and maintaining a copy of the reference model for KL computation incurs a compute cost we find unjustified. We remove the KL penalty entirely.

**Loss normalization.** To avoid introducing length biases between generations in one group, we normalize the loss by first adding token-wise loss for all tokens and all generations and then dividing by the total length of generations in the group  $\sum_{i=1}^G |o_i|$ .

**Advantage normalization.** We estimate the advantage of each token simply as  $\hat{A}_{i,t} = \hat{A}_i = r_i - \mu$ , where  $\mu$  is the mean of rewards within a group. Following [Andrychowicz et al., 2020], we additionally normalize the advantages in each minibatch as  $\hat{A}_{i,t}^{\text{norm}} = (\hat{A}_i - \hat{A}^{\text{mean}}) / \hat{A}^{\text{std}}$  where  $\hat{A}^{\text{mean}}$  and  $\hat{A}^{\text{std}}$  are the sequence-wise mean and standard deviation of the advantages  $\hat{A}_i$  in a minibatch.

**Relaxing the trust region’s upper bound.** We allow the model to explore rare but potentially insightful reasoning steps, preventing deterministic policies. We adopt the *Clip-Higher* [Yu et al., 2025] strategy to address entropy collapse. In standard GRPO,  $\varepsilon$ -clipping limits exploration by restricting the increase in probability of low-likelihood tokens, hindering the reinforcement of rare but important reasoning paths. By increasing the upper clipping threshold to  $\varepsilon_{\text{high}}$ , low-probability tokens have more room to grow, enhancing entropy and diversity in outputs, and improving reasoning exploration. We found that careful tuning of  $\varepsilon_{\text{high}}$  is crucial to maintaining stability in the RL run. We adjusted it between 0.26 and 0.28 during the training to keep the group entropy stable.

**Eliminating non-diverse groups.** Groups where all generations are either entirely correct or wrong have zero advantage and therefore contribute nothing to the batch loss. This results in smaller gradients with increased noise sensitivity. To address this, we filter out all groups with zero advantage when forming training batches.

The final GRPO loss with all modifications highlighted in red is

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}_{q \sim P(Q), \{o_i\}_{i=1}^G \sim \pi_{\theta_{\text{old}}}(\cdot|q)} \frac{1}{\sum_{i=1}^G |o_i|} \sum_{i=1}^G \sum_{t=1}^{|o_i|} \min \left[ \frac{\pi_{\theta}(o_{i,t}|q, o_{i,<t})}{\pi_{\theta_{\text{old}}}(o_{i,t}|q, o_{i,<t})} \hat{A}_{i,t}^{\text{norm}}, \text{clip}\left(\frac{\pi_{\theta}(o_{i,t}|q, o_{i,<t})}{\pi_{\theta_{\text{old}}}(o_{i,t}|q, o_{i,<t})}, 1 - \varepsilon_{\text{low}}, 1 + \varepsilon_{\text{high}}\right) \hat{A}_{i,t}^{\text{norm}} \right],$$

s. t.  $\exists 1 \leq m < n \leq G, r_m \neq r_n$ .

## 2.2 Reward shaping

Choosing the appropriate reward is crucial for the RL algorithm to work effectively. During training, model generations are evaluated along four axes: formatting, correctness, length, and language consistency, which we describe below.

### 2.2.1 Formatting

For both math and code problems, we instruct the model to follow a specific format, which facilitates the extraction of the model’s answer:

1. **Tag requirements:** (i) The model response must start with a `<think>` tag and must include a corresponding `</think>` tag. (ii) There should be exactly one set of these tags present in the response.
2. **Mathematical responses:** For mathematical outputs, the response must include the final answer enclosed in `\boxed{\}` within the answer section, following the `</think>` tag.
3. **Code responses:** For code outputs, the response must include at least one markdown block, formatted with triple backticks followed by the programming language specification, in the answer section.



在单个组内计算的奖励偏差。基于之前将GRPO应用于推理任务的工作 [Yu et al., 2025, Liu et al., 2025, Hu et al., 2025], 我们引入了几项修改:

消除KL散度。KL散度惩罚限制了在线策略偏离参考策略的程度, 有助于保持与初始模型的一致性。然而, 在GRPO中, 策略无论如何都会发生较大偏离, 且维护一份参考模型的副本以进行KL计算会带来我们认为不合理的计算成本。我们完全去除KL惩罚。

损失归一化。为了避免在同一组中不同生成之间引入长度偏差, 我们首先对所有标记和所有生成的标记损失进行求和, 然后再除以该组中所有生成的总长度  $\sum_{i=1}^G |o_i|$ 。

优势归一化。我们将每个标记的优势简单估计为  $\hat{A}_{i,t} = \hat{A}_i = r_i - \mu$ , 其中  $\mu$  是组内奖励的平均值。参考 Andrychowicz 等人 [2020], 我们还在每个小批次中对优势进行归一化, 公式为  $\hat{A}_{i,t}^{\text{norm}} = (\hat{A}_i - \hat{A}^{\text{mean}}) / \hat{A}^{\text{std}}$ , 其中  $\hat{A}^{\text{mean}}$  和  $\hat{A}^{\text{std}}$  分别是小批次中优势  $\hat{A}_i$  的序列均值和标准差。

放宽信任区域的上界。我们允许模型探索罕见但可能具有洞察力的推理步骤, 防止确定性策略。我们采用 *Clip-Higher* [Yu 等人, 2025] 的策略来应对熵崩溃。在标准的 GRPO 中,  $\varepsilon$  剪裁通过限制低概率标记的概率增加来限制探索, 阻碍了对罕见但重要的推理路径的强化。通过将上限剪裁阈值提高到  $\varepsilon_{\text{high}}$ , 低概率标记有更多空间增长, 增强了熵和输出的多样性, 并改善了推理探索。我们发现, 仔细调整  $\varepsilon_{\text{high}}$  对于在 RL 运行中保持稳定至关重要。在训练过程中, 我们将其调整在 0.26 和 0.28 之间, 以保持组熵的稳定。

消除非多样性组。所有世代都完全正确或错误的组没有优势, 因此对批次损失没有贡献。这会导致梯度变小, 噪声敏感性增加。为了解决这个问题, 我们在形成训练批次时过滤掉所有没有优势的组。

最终的GRPO损失, 所有修改用红色标出为

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}_{q \sim P(Q), \{o_i\}_{i=1}^G \sim \pi_{\theta_{\text{old}}}(\cdot|q)} \frac{1}{\sum_{i=1}^G |o_i|} \sum_{i=1}^G \sum_{t=1}^{|o_i|} \min \left[ \frac{\pi_{\theta}(o_{i,t}|q, o_{i,<t})}{\pi_{\theta_{\text{old}}}(o_{i,t}|q, o_{i,<t})} \hat{A}_{i,t}^{\text{norm}}, \text{clip}\left(\frac{\pi_{\theta}(o_{i,t}|q, o_{i,<t})}{\pi_{\theta_{\text{old}}}(o_{i,t}|q, o_{i,<t})}, 1 - \varepsilon_{\text{low}}, 1 + \varepsilon_{\text{high}}\right) \hat{A}_{i,t}^{\text{norm}} \right],$$

s. t.  $\exists 1 \leq m < n \leq G, r_m \neq r_n$ .

## 2.2 奖励塑造

选择合适的奖励对于强化学习算法的有效运行至关重要。在训练过程中, 模型生成的内容会沿着四个维度进行评估: 格式、正确性、长度和语言一致性, 具体如下所述。

### 2.2.1 格式化

对于数学和代码问题, 我们指导模型遵循特定的格式, 以便于提取模型的答案:

1. 标签要求: (i) 模型响应必须以 `<think>` 标签开头, 并且必须包含相应的 `</think>` 标签。 (ii) 响应中应恰好包含一组这些标签。
2. 数学响应: 对于数学输出, 响应必须在 `</think>` 标签之后的答案部分中包含用 `\boxed{\}` 括起来的最终答案。
3. 代码响应: 对于代码输出, 响应必须在答案部分包含至少一个用三重反引号括起来的代码块, 并指定编程语言。

Failure to meet any of these conditions results in a reward of 0, and the response will not be graded further. Otherwise, the response gets a reward of 0.1 and proceeds to grading.

### 2.2.2 Correctness

If the generated answer follows the required formatting, we extract the model solution and use a verifier to assess its correctness.

**Math correctness.** The final answer is extracted from inside the last `\boxed{\}` in the solution and compared against the reference answer using a rule-based verifier. It normalizes both the ground-truth and the generated answer to correctly reward semantically identical responses with different syntaxes. We leverage a combination of different parsers and SymPy [1] to evaluate outputs and compare them to the original ground truth. An additional reward of 0.9 is given if the answer is correct, making the total reward 1.0.

**Code correctness.** Code is extracted from the first markdown code block in the answer section. If the code is written in C++, it is compiled with a timeout of 10 seconds, using the C++20 standard. We pre-compile the `bits/stdc++.h` standard library header, which is commonly used in competitive programming, to speed up the compilation process. We randomly select 20 tests from the available test cases, ensuring that the same tests are used within a given response group. The code is then executed against these tests, with each test having a timeout of 4 seconds and a memory limit of 300 MB. An additional reward of 0.9 is given if the code successfully passes all the tests.

### 2.2.3 Length penalty

Following [Yu et al., 2025], we use soft length penalty to signal the model that the hard cutoff on maximal completion length is near. We fix two lengths  $l_{\max}$  and  $l_{\text{cache}}$  and compute length penalty as

$$R_{\text{length}}(y) = \begin{cases} 0, & |y| \leq l_{\max} - l_{\text{cache}} \\ -0.1 \cdot \frac{|y| - l_{\max} + l_{\text{cache}}}{l_{\text{cache}}}, & l_{\max} - l_{\text{cache}} < |y| \leq l_{\max}, \\ -0.1, & l_{\max} < |y| \end{cases} \quad (1)$$

### 2.2.4 Language consistency reward

A core design principle for Magistral is for it to reason in the same language as the user. Reinforcement learning on math and coding problems without any treatment often results in mixed-language model responses. In preliminary experiments without language constraints, we frequently observed outputs that mixed English, Chinese, and Russian words. While these outputs were coherent, they were undesirable from a user perspective.

To prevent language switching, we translated 10% of our problems written in English to the following languages: French, Spanish, Italian, German, Chinese, and Russian. When calculating the reward for a conversation—a triple of (problem, thoughts, answer)—we first normalized each of the three components by removing LaTeX content and code blocks, and then applied a fastText classifier [Joulin et al., 2016] to each. If the classifier indicates that all three parts used the same language, we give an additional reward of 0.1.

These simple modifications are sufficient to enable the model to closely follow the language of the user, with minimal code-switching, while maintaining performance on reasoning tasks. Although we only translated the original English problems into a few languages, we observed that the model could successfully generate chains of thought in arbitrary languages.

**System prompt.** We specify the format and the language requirements in the system prompt, which can be found in Figure 2. We find that RL training is quite sensitive to the system prompt we use. For example, the `Be as casual and as long as you want` part of the system prompt increases the entropy of the model and therefore improves the exploration of the model.



未能满足任何这些条件将导致奖励为0，且该回答不会被进一步评分。否则，回答将获得0.1的奖励并继续评分。

### 2.2.2 正确性

如果生成的答案符合所需的格式，我们将提取模型解答并使用验证器来评估其正确性。

数学正确性。最终答案从解答中的最后一个 `\boxed{}` 内提取，并使用基于规则的验证器与参考答案进行比较。它对真实答案和生成的答案进行归一化，以正确奖励语义相同但语法不同的回答。我们结合使用不同的解析器和 SymPy<sup>2</sup> 来评估输出，并将其与原始真实答案进行比较。如果答案正确，则额外奖励0.9，总奖励为1.0。

代码正确性。代码从回答部分的第一个 markdown 代码块中提取。如果代码是用 C++ 编写的，则使用 C++20 标准，在 10 秒的超时时间内进行编译。我们预先编译了 `bits/stdc++.h` 标准库头文件，这在竞赛编程中常用，以加快编译速度。我们从可用的测试用例中随机选择 20 个测试，确保在同一响应组内使用相同的测试。然后对这些测试运行代码，每个测试的超时时间为 4 秒，内存限制为 300 MB。如果代码成功通过所有测试，将额外获得 0.9 的奖励。

### 2.2.3 长度惩罚

遵循 [Yu 等人, 2025]，我们使用软长度惩罚来提示模型，最大完成长度的硬截止点即将到来。我们固定两个长度  $l_{\max}$  和  $l_{\text{cache}}$ ，并计算长度惩罚为

$$R_{\text{length}}(y) = \begin{cases} 0, & |y| \leq l_{\max} - l_{\text{cache}} \\ -0.1 \cdot \frac{|y| - l_{\max} + l_{\text{cache}}}{l_{\text{cache}}}, & l_{\max} - l_{\text{cache}} < |y| \leq l_{\max}, \\ -0.1, & l_{\max} < |y| \end{cases} \quad (1)$$

### 2.2.4 语言一致性奖励

Magistral 的核心设计原则是让它用与用户相同的语言进行推理。在没有任何限制的情况下对数学和编码问题进行强化学习，通常会导致模型输出混合语言的回答。在没有语言限制的初步实验中，我们经常观察到输出中混杂英语、中文和俄语的词汇。虽然这些输出是连贯的，但从用户的角度来看是不理想的。

为了防止切换语言，我们将用英语编写的10%的问题翻译成了以下语言：法语、西班牙语、意大利语、德语、中文和俄语。在计算一次对话的奖励时——即（问题、思考、回答）三元组——我们首先通过去除LaTeX内容和代码块对这三个部分进行归一化，然后对每个部分应用fastText分类器 [Joulin et al., 2016]。如果分类器显示这三部分都使用了相同的语言，我们会额外给予0.1的奖励。

这些简单的修改足以使模型能够紧密跟随用户的语言，几乎不需要代码切换，同时保持在推理任务上的性能。虽然我们只将原始的英文问题翻译成了几种语言，但我们观察到模型可以在任意语言中成功生成思维链。

系统提示。我们在系统提示中指定了格式和语言要求，详见图2。我们发现，强化学习训练对所使用的系统提示非常敏感。例如，系统提示中的“尽可能随意且长”部分会增加模型的熵，从而改善模型的探索能力。

### Magistral’s system prompt

A user will ask you to solve a task. You should first draft your thinking process (inner monologue) until you have derived the final answer. Afterwards, write a self-contained summary of your thoughts (i.e. your summary should be succinct but contain all the critical steps you needed to reach the conclusion). You should use Markdown and Latex to format your response. Write both your thoughts and summary in the same language as the task posed by the user.

Your thinking process must follow the template below:

<think>

Your thoughts or/and draft, like working through an exercise on scratch paper. Be as casual and as long as you want until you are confident to generate a correct answer.

</think>

Here, provide a concise summary that reflects your reasoning and presents a clear final answer to the user.

Problem:

{problem}

**Figure 2: Magistral’s system prompt.** The system prompt spells out the format and language guidelines for the model. The same system prompt is utilized for both mathematical and coding problems.

## 3 Infrastructure

In this section, we present our infrastructure for online training. We adopt a distributed RL training system similar to those proposed in several prior works [Espeholt et al., 2018, Hu et al., 2024, Noukhovitch et al., 2024, Sheng et al., 2024, Wu et al., 2025] that coordinates three kinds of workers:

- **Trainers** maintain the main copy of the model weights and perform gradient updates.
- **Generators** perform ‘roll-outs’, using the latest policy to return completions with log-probabilities from the training prompts.
- **Verifiers** evaluate the completions produced by the generators and return a reward (see Section 2.2 for details).

**Challenges with distributed RL.** Generators are a significant part of the total compute and the part that’s unique to online RL. Their workload is highly heterogeneous and hard to predict as the distribution of sequence lengths is highly skewed and changes over the course of training: the longest completions can take up to 5 times longer than the shortest. One of the main constraints of the system is to introduce no bias on sequence lengths: the distribution of completion lengths must be exactly that of the training data, even though shorter completions finish more quickly. A competing goal is to update the generator weights as soon as possible. We want the generations to be as on-policy as possible, but we also want the generators to operate without waiting for each other or the trainers.

**Asynchronous generations.** In order to train without any approximation, we could process batches sequentially: start generators on a batch, wait for all sequences to complete, update the model weights for both trainers and generators, and repeat. However, this approach leads to idle generators and low pipeline efficiency due to heterogeneous completion times. Instead, we prioritize efficiency and operate the generators continuously at maximum throughput without ever waiting for the trainers. We constantly gather groups from the generators, verify them, and update the trainers. After these updates, the trainers send the new weights to the generators via NCCL, without discarding the in-flight sequences currently being generated. Broadcasting weights from GPUs to GPUs is crucial as it reduces the time required for a single update to below 5 seconds, even with large models and large world sizes. We illustrate this process in Figure 3.

<sup>2</sup><https://www.sympy.org/en/index.html>

### Magistral的系统提示

一位用户会要求你解决一个任务。你应该首先草拟你的思考过程（内心独白），直到你得出最终答案。之后，写一个自包含的思考总结（即你的总结应简洁但包含所有关键步骤，以达到结论）。你应使用Markdown和Latex来格式化你的回答。用与用户提出任务相同的语言书写你的思考和总结。

Your thinking process must follow the template below: 你的思考过程必须遵循以下模板：  
<think> 你的想法或/和草稿，就像在草稿纸上做练习一样。可以随意、尽可能长地写，直到你有信心生成正确的答案。</think>

在这里，提供一个简洁的总结，反映你的推理过程，并向用户呈现一个明确的最终答案。

Problem:问题：

{problem}

图2: Magistral的系统提示。系统提示详细说明了模型的格式和语言指南。对于数学问题和编码问题，使用相同的系统提示。

## 3 基础设施

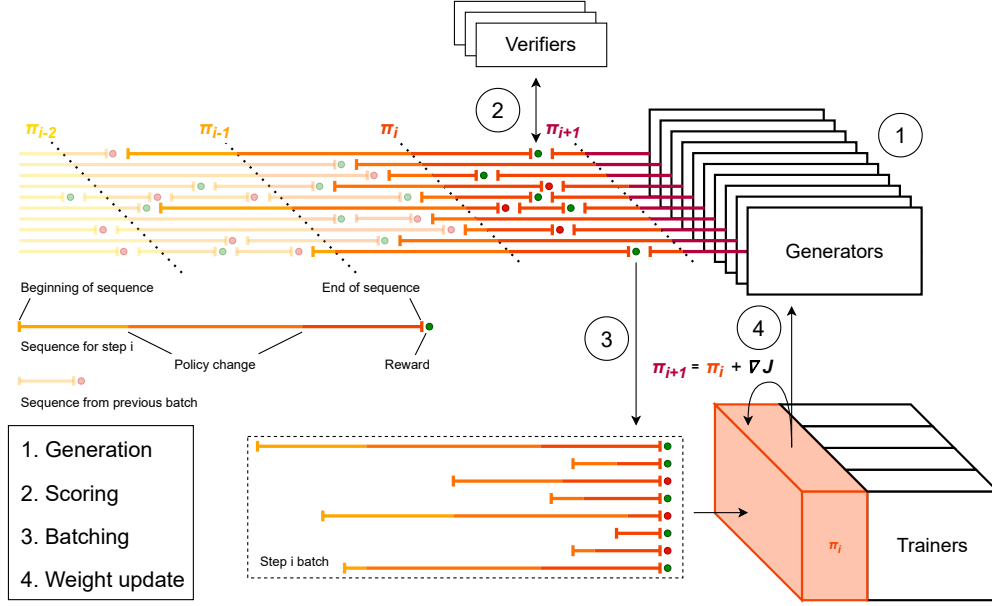
在本节中，我们介绍我们的在线训练基础设施。我们采用一种类似于之前多项工作[Espeholt 等, 2018年, Hu 等, 2024年, Noukhovitch 等, 2024年, Sheng 等, 2024年, Wu 等, 2025年]提出的分布式强化学习训练系统，该系统协调三种类型的工作者：

- 训练师维护模型权重的主副本并执行梯度更新。
- 生成器执行“滚动”，使用最新策略返回带有训练提示的对数概率的完成。
- 验证器评估生成器产生的完成内容并返回奖励（详见第2.2节）。

分布式强化学习的挑战。生成器是总计算量的重要部分，也是在线强化学习中特有的部分。它们的工作负载高度异质且难以预测，因为序列长度的分布高度偏斜，并且在训练过程中不断变化：最长的完成可能比最短的完成长达5倍。系统的主要限制之一是不能在序列长度上引入偏差：完成长度的分布必须与训练数据的分布完全一致，即使较短的完成可以更快完成。另一个竞争目标是尽快更新生成器的权重。我们希望生成尽可能符合策略，但也希望生成器能够在不等待彼此或训练者的情况下运行。

异步生成。为了实现无近似的训练，我们可以按顺序处理批次：启动生成器处理一个批次，等待所有序列完成，更新两个训练器和生成器的模型权重，然后重复。然而，这种方法会导致生成器空闲和管道效率低下，因为完成时间异质。相反，我们优先考虑效率，持续以最大吞吐量运行生成器，从不等待训练器。我们不断从生成器收集组，验证它们，并更新训练器。在这些更新之后，训练器通过NCCL将新权重发送给生成器，而不丢弃当前正在生成的飞行序列。从GPU到GPU广播权重至关重要，因为它可以将单次更新所需的时间降低到5秒以下，即使是在大型模型和大规模世界中。我们在图3中说明了这个过程。

<sup>2</sup><https://www.sympy.org/en/index.html>



**Figure 3: Online training pipeline.** 1) Generators continuously output completions to prompts from input data sources. 2) Whenever a completion is finished, it is sent to the appropriate verifier. 3) Each sequence is sent to a different data parallel group using a pre-set permutation until every data parallel group has enough sequences to form a batch. 4) A single gradient step is performed and the trainer and generators are updated. In the generators, weights are replaced mid-generation, which means that in-flight generations continue with a slightly outdated key-value cache, as we do not refresh the cache. Since the model resides on GPUs in both the trainer and the generators, the weights are transferred using NCCL for optimal performance. The model weights are dynamically consolidated to accommodate the different sharding topologies between trainers and generators.

As a solution is generated for a single prompt, it may experience multiple updates to the model weights, reflecting the latest improvements from the trainers. By the time it is fully processed and sent to the trainers, the model weights may have been updated several times, but the latest tokens are always generated on-policy. When updating the model weights, the hidden states previously stored in the key-value cache become slightly outdated because they were computed by previous versions of the model. For performance, we find that recomputing the key-value cache is not necessary, potentially due to off-policy corrections inherent to the loss function [Schulman et al., 2017].

**Trainer optimization.** We define a batch as a fixed number of generated completions, rather than a fixed number of tokens. Generators send each finished completion to a random trainer rank according to a pre-set permutation. A gradient update is performed when each data parallel rank has received enough completions to make a batch. If the trainers are the bottleneck, as is the case in early training when the generations are still short, we accumulate incoming generations into a blocking queue with a fixed size limit that controls off-policy degree. A batch may be partitioned into minibatches to perform several optimization steps (see Section 6.3). Each minibatch has a fixed number of completions but a variable number of tokens, so it is further divided into microbatches of a fixed token size. Since we accumulate the gradient over microbatches, the order of samples does not matter. We take advantage of this property to implement a greedy collation algorithm, sorting the sequences by descending size and trying to fit them one by one into a free microbatch if there is one or starting a new otherwise. This ensures a homogeneous workload across training workers for each minibatch, reducing padding by 19%.

## 4 Data curation

We limit ourselves to problems with verifiable solutions; we use mathematical problems whose solution is a numerical answer or expression, and code problems with associated tests. We apply extensive filtering, which we describe here.

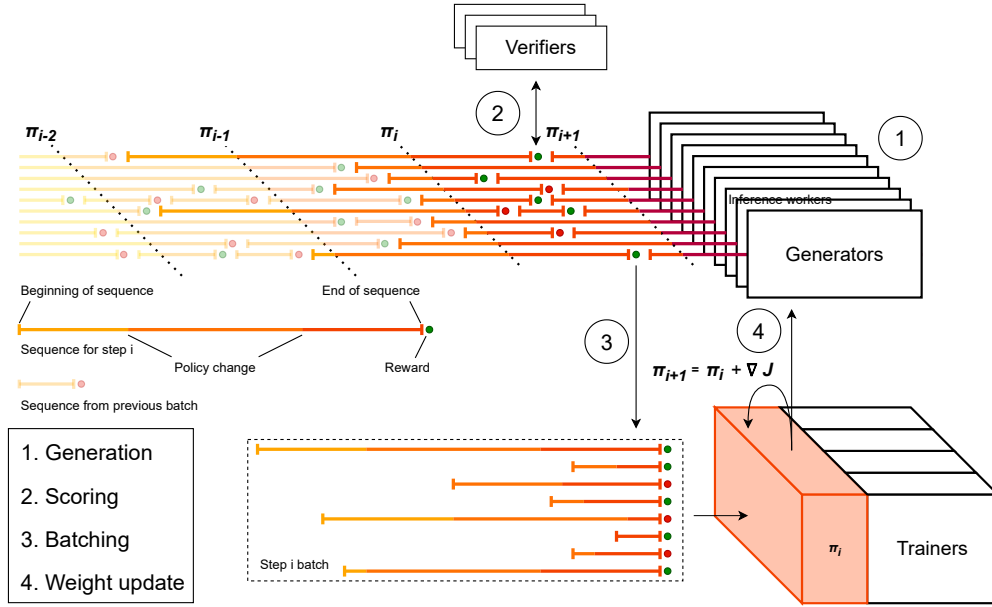


图3：在线训练流程。1) 生成器持续输出对输入数据源提示的完成内容。2) 每当完成内容生成后，便将其发送到相应的验证器。3) 每个序列使用预设的置换方式发送到不同的数据并行组，直到每个数据并行组拥有足够的序列以形成一个批次。4) 执行一次梯度更新，更新训练器和生成器。在生成器中，权重在生成过程中被替换，这意味着在飞行中的生成会继续使用略微过时的键值缓存，因为我们不刷新缓存。由于模型同时驻留在训练器和生成器的GPU上，权重通过NCCL进行传输以实现最佳性能。模型权重会动态整合，以适应训练器和生成器之间不同的分片拓扑结构。

当为单个提示生成解决方案时，它可能会多次更新模型权重，反映训练者的最新改进。在它完全处理完毕并发送给训练者之前，模型权重可能已经被多次更新，但最新的标记总是基于策略生成的。在更新模型权重时，之前存储在键值缓存中的隐藏状态会变得略微过时，因为它们是由模型的早期版本计算得出的。为了提高性能，我们发现不必重新计算键值缓存，可能是由于损失函数中固有的离策略校正 [Schulman 等, 2017]。

训练器优化。我们将一个批次定义为一组固定数量的生成完成，而不是固定数量的标记。生成器将每个完成的生成发送到一个随机的训练器节点，按照预设的排列。每当每个数据并行节点接收到足够的完成以组成一个批次时，就会进行一次梯度更新。如果训练器成为瓶颈，就像在早期训练时生成还很短的情况下，我们会将传入的生成积累到一个具有固定大小限制的阻塞队列中，以控制偏离策略的程度。一个批次可以被划分成多个微批次，以执行多次优化步骤（见第6.3节）。每个微批次具有固定数量的完成，但标记数量可变，因此会被进一步划分成固定标记数的微批次。由于我们在微批次上累积梯度，样本的顺序并不重要。我们利用这一特性实现了一种贪心拼接算法，将序列按降序排序，并尝试将它们逐一放入空闲的微批次中，如果没有空闲的则开启一个新的微批次。这确保了每个微批次在训练工作者之间具有均匀的工作负载，减少了19%的填充。

## 4 数据整理

我们将自己限制在具有可验证解决方案的问题上；我们使用解答为数值或表达式的数学问题，以及带有相关测试的编码问题。我们应用广泛的筛选方法，详细描述如下。

## 4.1 Math

**Format filtering.** We started with a large but noisy problem set of around 700k samples. We first perform comprehensive pre-processing and filtering of the data to ensure all the problems are complete and that the final answers were accurate and verifiable with a rule-based system. Particularly, we filter proof-based and multi-part problems for which it is difficult to verify correctness. Furthermore, we reformulate multiple-choice problems into statement-based problems for more robust verification and increased difficulty.

**Difficulty filtering.** We implemented a two-stage filtering pipeline to curate a dataset of problems at a ‘goldilocks’ difficulty level, neither too easy nor too hard for the model to learn from. First, we performed an initial difficulty assessment using Mistral Large 2 [MistralAI, 2024], by sampling 16 solutions for each problem and removing the ones that are either never solved or solved with a high success rate. This initial, curated set of problems was then used to train a 24B model via our online RL pipeline, resulting in a small but capable checkpoint which we use solely for grading.

In the second stage, this stronger, RL-trained model was used to re-grade the entire original dataset. We again sampled 16 responses for each problem, filtering out the easiest and the still-unsolved problems. We then further filter out potentially incorrect problems where a majority of samples have the same final answer but disagree with the “ground-truth” answer. This is because when the model consistently reaches a consensus that contradicts the reference solution, the problems themselves are more likely to have wrong ground-truth answers.

This two-stage methodology was crucial because a single pass with the initial, weaker Mistral Large 2 model would have been insufficient. Its reasoning capabilities would likely have caused it to discard many genuinely difficult problems by incorrectly classifying them as unsolvable. By using a stronger, RL-trained model for a second pass, we ensured that these valuable and challenging training examples were accurately assessed and retained.

**Table 1:** Number of math training samples after different filtering stages.

Initial data	w/ Format filtering	w/ Difficulty filtering
699k	501k	38k

## 4.2 Code

We gathered code contest data from various sources. Each data point includes a problem statement and, when available, correct solutions and related tests. For the training process, we want problem statements and a large number of correct tests per problem. In order to achieve this, we first remove any problems without solutions and without enough tests. Each solution is then executed on all available tests, and we discard tests with insufficient agreement. For tests with sufficient agreement but where no solution succeeded, we assume that the test is incorrect and update it to reflect the most common result among the solutions’ outputs. In cases where code problems lack tests, we generate additional tests and subject them to the same evaluation process.

Finally, where applicable, problem statements are duplicated to require code in Python or C++, two commonly used languages in competitive programming. This process resulted in a dataset of 35k code problems.

## 5 Experiment and results

In this section we present the Magistral models. Our goal is to answer two questions: (i) how far can one get with pure reinforcement learning on a large base model? (ii) given a strong teacher model, how can one achieve the strongest possible lightweight model? To this end, we trained Magistral Medium, on top of Mistral Medium 3 [MistralAI, 2025] with pure RL; and Magistral Small, which began with SFT traces derived from Magistral Medium.



## 4.1 数学

格式过滤。我们最初拥有一个庞大但噪声较多的问题集，约有70万条样本。我们首先对数据全面的预处理和过滤，以确保所有问题完整，并且最终答案是准确且可以通过基于规则的系统验证的。特别是，我们过滤了难以验证正确性的证明型和多部分问题。此外，我们将多项选择题重新表述为陈述型问题，以实现更稳健的验证和增加难度。

过滤难度。我们实现了一个两阶段的过滤流程，以策划一个“金发姑娘”难度水平的问题数据集，既不太容易也不太难，模型可以从中学习。首先，我们使用 Mistral Large 2 [MistralAI, 2024] 进行了初步的难度评估，通过对每个问题抽取 16 个解答样本，并移除那些从未解决或成功率很高的解答。这个初步筛选的题目集随后被用来通过我们的在线强化学习流程训练一个 24B 模型，从而得到一个小而强大的检查点，我们仅用它来进行评分。

在第二阶段，使用这个更强的、经过强化学习训练的模型对整个原始数据集进行了重新评分。我们再次为每个问题抽取了16个回答，过滤掉最容易的问题和仍未解决的问题。然后，我们进一步筛选出可能存在错误的问题，即大多数样本的最终答案相同但与“真实答案”不一致的情况。这是因为当模型持续达成与参考解答相矛盾的共识时，问题本身更可能存在错误的“真实答案”。

这种两阶段的方法至关重要，因为仅用初始较弱的 Mistral Large 2 模型进行一次处理是不够的。它的推理能力可能会导致它错误地将许多真正困难的问题判定为无法解决，从而丢弃这些宝贵且具有挑战性的训练样本。通过使用更强的、经过强化学习训练的模型进行第二次处理，我们确保了这些有价值且具有挑战性的训练示例能够被准确评估和保留。

表1：不同筛选阶段后的数学训练样本数量。

Initial data	w/ Format filtering	w/ Difficulty filtering
699k	501k	38k

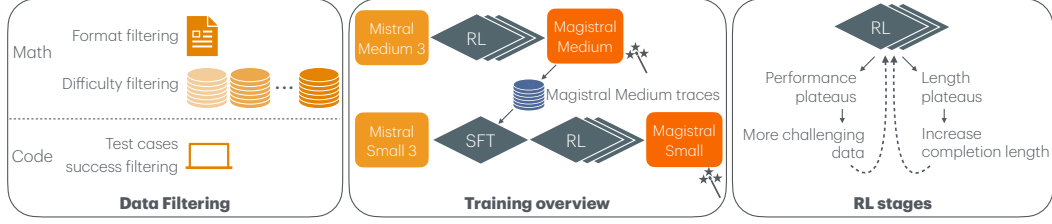
## 4.2 代码

我们从各种来源收集了代码竞赛数据。每个数据点包括一个问题陈述，以及在可用时的正确解和相关测试。为了训练过程，我们需要问题陈述和每个问题的大量正确测试。为此，我们首先删除没有解答或测试不足的问题。然后，将每个解答在所有可用测试上执行，并丢弃测试中不一致的部分。对于那些一致性足够但没有解答成功的测试，我们假设该测试是错误的，并将其更新为反映解答输出中最常见的结果{v\*}。在没有测试的代码问题情况下，我们会生成额外的测试，并对它们进行相同的评估过程。

最后，在适用的情况下，问题陈述被复制以要求用Python或C++编写代码，这两种语言在竞赛编程中被广泛使用。这个过程产生了一个包含3.5万道题目的数据集。

## 5 实验与结果

在本节中，我们介绍了Magistral模型。我们的目标是回答两个问题：(i) 仅靠纯强化学习，在一个大型基础模型上能达到多远？(ii) 给定一个强大的教师模型，如何实现最强的轻量级模型？为此，我们在Mistral Medium 3 [MistralAI, 2025]的基础上，使用纯RL训练了Magistral Medium；以及从Magistral Medium派生的SFT轨迹开始，训练了Magistral Small。



**Figure 4: Overview of the filtering, training and RL stages discussed in the paper.** We do RL over Mistral Medium 3 to get Magistral Medium. We use this model to generate answers for a large set of diverse prompts. We use these generated traces to finetune Mistral Small 3 and then perform RL to get Magistral Small.

## 5.1 Evaluation benchmarks and baselines

We report results on benchmarks that assess capabilities in the fields of mathematics, coding, and STEM. For math, we provide results on the American Invitational Mathematics Examination benchmarks (AIME’24, AIME’25), and on the MATH dataset [Hendrycks et al., 2021]. In coding, we include LiveCodeBench (both v5 and v6 versions) [Jain et al., 2024], and Aider Polyglot [Gauthier, 2024]. For STEM, we report results based on the GPQA dataset [Rein et al., 2024]. Additionally, we also report our results on the text-only questions from Humanity’s Last Exam [Phan et al., 2025] which comprises of 2,500 questions across dozens of subjects, including mathematics, humanities, and natural sciences. For all evaluation tasks, we set the temperature to 0.7 and use a top-p of 1.0 for Math evals and GPQA, and 0.95 for coding tasks. The maximum token length is set to 40k for AIME and LiveCodeBench, and 32k for all other evaluations.

For baselines we include results from [DeepSeek-AI et al., 2025], which reports comparable datapoints for training with RL at scale, both with and without SFT on traces from a reasoning model.

## 5.2 Magistral Medium – reasoning RL from scratch

Here our goal is to evaluate the quality of our RL stack by training a model without any ‘cold start’ (i.e priming for reasoning by distillation of reasoning traces). We used Mistral Medium 3 Instruct [MistralAI, 2025] as the starting checkpoint for this run. Training was done in multiple stages with distinct hyper-parameters. Particularly, the stages were designed to ensure the following criteria were always satisfied:

1. **Dataset is not too easy.** As the model performance increases, we increase the difficulty of the data. Harder data splits are constructed by including more complicated data (which were filtered out in earlier stages) or removing completely solved problems from the data.
2. **Generation length does not stop growing.** To prevent stagnation in generation length, we increase both maximal allowed completion length and maximal completion length  $l_{\max} - l_{\text{cache}}$  not punished by length penalty (c.f. Section 2.2.3). We increased  $l_{\max} - l_{\text{cache}}$  twice as  $16k \rightarrow 24k$  and  $24k \rightarrow 32k$ .
3. **KV-cache memory burden is not too large.** As generation length increases, the memory usage associated with the KV cache increases. To address this, we scale down the total number of concurrent requests running  $n_{\text{async}}$ , the batch size  $n_{\text{batch}}$ , and the minibatch size  $n_{\text{minibatch}}$ . The impact of batch size is discussed in Section 6.3. During training we decreased batch size twice as  $8k \rightarrow 4k$  and  $4k \rightarrow 2k$ .

Table 2 shows the results of Magistral Medium trained with pure RL, compared against analogous experiments from [DeepSeek-AI et al., 2025]. We find our RL pipeline alone yields a nearly 50% accuracy increase in AIME ’24 (pass@1), and 30% on LiveCodeBench (v5).

## 5.3 Magistral Small – RL on top of reasoning SFT bootstrapping

Given a strong ‘teacher’ model in Magistral Medium, we next explore how one can train the strongest possible student model. To do so, we train Magistral Small, which is ‘cold-started’ with SFT traces from Magistral Medium.

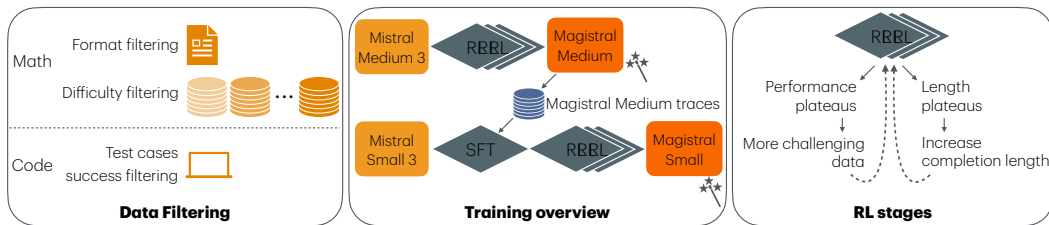


图4: 本文讨论的过滤、训练和强化学习 (RL) 阶段的概述。我们在 Mistral Medium 3 上进行 RL，以获得 Magistral Medium。我们使用该模型为大量多样化的提示生成答案。然后，我们利用这些生成的轨迹对 Mistral Small 3 进行微调，之后进行 RL 以获得 Magistral Small。

### 5.1 评估基准和基线

我们报告了在数学、编码和STEM领域能力评估基准上的结果。在数学方面，我们提供了美国邀请赛数学考试 (AIME' 24、AIME' 25) 以及MATH数据集 [Hendrycks et al., 2021] 的结果。在编码方面，我们包括了LiveCodeBench (v5和v6版本) [Jain et al., 2024] 和Aider Pol yglot [Gauthier, 2024]。在STEM方面，我们基于GPQA数据集 [Rein et al., 2024] 报告了结果。此外，我们还报告了在《人类的最后考试》 [Phan et al., 2025] 中的纯文本问题的结果，该考试包含来自数十个学科的2,500个问题，包括数学、人文学科和自然科学。对于所有评估任务，我们将温度设置为0.7，数学评估和GPQA使用top-p为1.0，编码任务使用top-p为0.95。AIME和LiveCodeBench的最大令牌长度设置为40k，其他所有评估的最大令牌长度为32k。

对于基线，我们包括了 [DeepSeek-AI 等人, 2025] 的结果，该结果报告了在大规模使用强化学习进行训练的可比数据点，既包括有SFT也包括无SFT的推理模型轨迹。

### 5.2 主要媒介 – 从零开始推理强化学习 {v\*}

在这里，我们的目标是通过训练一个没有任何“冷启动”（即通过推理轨迹的蒸馏来引导推理）的模型来评估我们的强化学习堆栈的质量。我们使用 Mistral Medium 3 Instruct [MistralAI, 2025] 作为此次运行的起始检查点。训练分为多个阶段，采用不同的超参数。特别地，这些阶段的设计旨在确保始终满足以下标准：

1. 数据集并不太容易。随着模型性能的提升，我们会增加数据的难度。更难的数据划分通过包含更复杂的数据（在早期阶段被过滤掉的）或完全移除已解决的问题来构建。
2. 生成长度不会停止增长。为了防止生长度的停滞，我们同时增加最大允许完成长度和不受长度惩罚限制的最大完成长度  $l_{\max} - l_{\text{cache}}$ （参见第2.2.3节）。我们将  $l_{\max} - l_{\text{cache}}$  增加到原来的两倍，达到16k  $\rightarrow$  24k 和 24k  $\rightarrow$  32k。
3. KV缓存内存负担不太大。随着生长度的增加，与KV缓存相关的内存使用也会增加。为了解决这个问题，我们缩减了同时运行的请求总数  $n_{\text{async}}$ 、批处理大小  $n_{\text{batch}}$  和小批量大小  $n_{\text{minibatch}}$ 。批处理大小的影响在第6.3节中讨论。在训练过程中，我们将批处理大小减半，达到8k  $\rightarrow$  4k 和 4k  $\rightarrow$  2k。

表2显示了使用纯RL训练的Magistral Medium的结果，与 [DeepSeek-AI等, 2025] 的类似实验进行了比较。我们发现，仅靠我们的RL流程在AIME' 24 (pass@1) 中就实现了近50%的准确率提升，在LiveCodeBench (v5) 中则提升了30%。

### 5.3 Magistral Small – 在推理SFT自举之上的RL

给定一个强大的“教师”模型在Magistral Medium中，我们接下来探索如何训练最强的学生模型。为此，我们训练Magistral Small，它是用Magistral Medium的SFT轨迹“冷启动”而来的。

In contrast with pure RL training (which benefits from a small set of extremely clean and difficult training points, Section 4), we find diversity of prompts to be important for the reasoning cold-start. We begin by extracting traces with correct answers from the RL training of Magistral Medium, excluding those from early steps with short CoTs. We also maintain a mixed difficulty level of the problems by limiting number of generations per problem to avoid biasing the collected traces towards easier problems and also upsampling problems with lower pass rates.

We augment this SFT cold-start data by generating responses from our Magistral Medium on a large set of diverse prompts, sourced from OpenThoughts [Guha et al., 2025] and the code subset of OpenR1 [Hugging Face, 2025, Penedo et al., 2025]. We perform additional filtering on top and kept a subset of the prompts. This gives us a reasoning dataset with mixed difficulty. We also include 10% of datapoints for general instruction tuning in order to preserve non-reasoning capabilities. We finetuned Mistral Small 3 Instruct (a 24-billion parameter model) for 4 epochs, and chose the best checkpoint on AIME’24 as the initial checkpoint for the following RL stage.

We then trained this SFT checkpoint with RL using a batch size of 2048 sequences, and a maximum non-penalized completion length  $l_{\max} - l_{\text{cache}}$  of 32k. We used a sampling temperature of 1.0 for our generations, as it provided the best balance between avoiding the lack of diversity seen at lower temperatures and the incoherent outputs generated at higher temperatures. We use a  $\varepsilon_{\text{high}}$  of 0.3, to encourage exploration, as the cold-started model yielded responses with far lower entropy.

Table 3 shows the performance of the 24B model trained under three different paradigms: with SFT alone; with RL alone; and with RL on top of the cold-start checkpoint. Here, contrary to findings from [DeepSeek-AI et al., 2025], we find one can get substantial boosts with RL even on a smaller base model, over and above distillation from the larger teacher. This underscores the strength of the RL stack introduced in this work.

**Table 2: Results of Magistral Medium trained solely with RL.** To reduce variance, we compute the average over 64 runs for AIME (shown as pass@1/maj@64) and over 16 runs for LiveCodeBench. Humanity’s Last Exam is evaluated only for the text subset.

Task	Mistral Medium 3	Magistral Medium	DeepSeek- v3	DeepSeek- R1-Zero	DeepSeek- R1
Reasoning SFT before RL	-	✗	-	✗	✓
AIME’24	26.8 / 43.4	73.6 / 90.0	39.2	71.0	79.8
AIME’25	21.2 / 30.0	64.9 / 83.3	28.8	-	70.0
MATH-500	91.0	94.3	90.2	95.9	97.3
GPQA	59.6	70.8	59.1	73.3	71.5
LiveCodeBench (v5)	29.1	59.4	36.2	50.0	65.9
Aider Polyglot	28.9	47.1	49.6	-	53.3
LiveCodeBench (v6)	30.0	50.3	-	-	-
Humanity’s Last Exam	4.4	9.0	-	-	8.6

## 5.4 Multilingual benchmarks

To evaluate Magistral’s multilingual capabilities, we interacted with Magistral Medium in multiple languages to check that it could reason and answer in the user’s language. We also tested Magistral Medium on multilingual (French, Spanish, German, Italian, Russian, and Chinese) versions of the AIME 2024 benchmark. These multilingual versions were created by translating the questions from English into each of the languages. The results are presented in Table 4. We see that the model performs 4.3-9.9% lower on multilingual versions compared to English, which corresponds to 1-3 questions on the actual AIME test, possibly because we constrained the language of reasoning. This degradation is roughly similar to that of the base model. Note that on the multilingual benchmarks, all of the reasoning and the final response are conducted in the input language (i.e., not English).

与纯粹的RL训练（受益于一小部分极其干净且困难的训练点，见第4节）相比，我们发现提示的多样性对于推理的冷启动非常重要。我们首先从Magistral Medium的RL训练中提取带有正确答案的轨迹，排除那些来自早期步骤且包含短链的CoT的轨迹。我们还通过限制每个问题的生成次数，保持问题的难度水平混合，以避免偏向较易的问题，同时对通过率较低的问题进行过采样。

我们通过从我们的Magistral Medium对一组多样化的提示中生成响应，来增强这个SFT冷启动数据，这些提示来自OpenThoughts [Guha等，2025]和OpenR1 [Hugging Face, 2025, Penedo等，2025]的代码子集。我们在此基础上进行了额外的筛选，并保留了一部分提示。这为我们提供了一个难度混合的推理数据集。我们还包括了10%的数据点用于一般指令调优，以保持非推理能力。我们对Mistral Small 3 Instruct（一个拥有240亿参数的模型）进行了4轮微调，并选择了AIME’24上的最佳检查点作为后续RL阶段的初始检查点。

然后，我们使用RL对这个SFT检查点进行了训练，批量大小为2048个序列，最大非惩罚性完成长度 $l_{\max} - l_{\text{cache}}$ 为32k。我们在生成时采用了温度为1.0的采样，因为它在避免低温下缺乏多样性和高温下生成不连贯输出之间提供了最佳平衡。我们使用 $\epsilon_{\text{high}}$ 为0.3，以鼓励探索，因为冷启动模型产生的响应熵要低得多。

表3显示了在三种不同范式下训练的24B模型的性能：仅使用SFT；仅使用RL；以及在冷启动检查点基础上进行RL。在这里，与[DeepSeek-AI等，2025]的发现相反，我们发现即使在较小的基础模型上，使用RL也能获得显著的提升，超出从更大教师模型进行蒸馏的效果。这突显了本文引入的RL堆栈的优势。

表2：仅使用RL训练的Magistral Medium的结果。为了减少方差，我们对AIME的64次运行取平均（显示为pass@1/maj@64），对LiveCodeBench的16次运行取平均。Humanity’s Last Exam仅在文本子集上进行评估。

Task	Mistral Medium 3	Magistral Medium	DeepSeek-v3	DeepSeek-R1-Zero	DeepSeek-R1
Reasoning SFT before RL	-	$\times$	-	$\times$	✓
AIME’24	26.8 / 43.4	73.6 / 90.0	39.2	71.0	79.8
AIME’25	21.2 / 30.0	64.9 / 83.3	28.8	-	70.0
MATH-500	91.0	94.3	90.2	95.9	97.3
GPQA	59.6	70.8	59.1	73.3	71.5
LiveCodeBench (v5)	29.1	59.4	36.2	50.0	65.9
Aider Polyglot	28.9	47.1	49.6	-	53.3
LiveCodeBench (v6)	30.0	50.3	-	-	-
Humanity’s Last Exam	4.4	9.0	-	-	8.6

#### 5.4 多语言基准测试

为了评估Magistral的多语言能力，我们用多种语言与Magistral Medium进行了互动，以检查它是否能够用用户的语言进行推理和回答。我们还在多语言（法语、西班牙语、德语、意大利语、俄语和中文）版本的AIME 2024基准测试中测试了Magistral Medium。这些多语言版本是通过将问题从英语翻译成每种语言而创建的。结果显示在表4中。我们发现，该模型在多语言版本上的表现比英语版本低4.3-9.9%，这相当于实际AIME测试中的1-3个问题，可能是因为我们限制了推理的语言。这种性能下降大致与基础模型的表现相似。请注意，在多语言基准测试中，所有的推理和最终回答都是用输入语言（即非英语）进行的。

**Table 3: Performance of Magistral Small compared with different training setups across various benchmarks.** We report the performance of three distinct 24B models: Mistral Small 24B fine-tuned on reasoning traces from Magistral Medium (SFT), Mistral Small 24B trained from scratch with RL (RL only), and Mistral Small 24B fine-tuned on Magistral Medium traces and subsequently enhanced with RL (SFT + RL) which is the final Magistral Small. We observe that the combination of fine-tuning on reasoning traces with RL leads to the best performance. For the evaluation of Humanity’s Last Exam, only the text subset was considered.

Task	SFT	RL-only	SFT + RL (Magistral Small)
AIME’24 <sub>pass@1</sub>	65.4	65.8	70.7
AIME’24 <sub>maj@64</sub>	90.0	86.7	83.3
AIME’25 <sub>pass@1</sub>	55.6	51.9	62.8
AIME’25 <sub>maj@64</sub>	76.7	66.7	76.7
MATH-500	93.2	95.4	95.9
GPQA	63.4	68.8	68.2
LiveCodeBench (v5)	52.2	46.4	55.8
LiveCodeBench (v6)	44.6	42.4	47.4
Humanity’s Last Exam	5.3	6.1	6.4

**Table 4:** Magistral Medium’s pass@1 performance on multilingual versions of the AIME 2024 benchmark.

Language	English	French	Spanish	German	Italian	Russian	Chinese
AIME’24 (pass@1)	73.6	68.5	69.3	66.8	66.7	65.0	63.7

## 6 Ablations

In this section, we tweak parameters of the training process to investigate what happens when RL is performed on only one modality, compare RL to the distillation SFT baseline, and shed light on two training choices we had to reckon with, batch and minibatch size, and advantage normalization.

### 6.1 Cross-domain generalization

We investigate the ability of our model to generalize across domains by training on one domain (math or code) and evaluating on the other. Specifically, we conduct two experiments on the 24B model: one where the model is trained exclusively on math data and evaluated on both math and code, and another where it is trained only on code and evaluated similarly. As shown in Table 5, the model demonstrates strong performance to out-of-domain tasks, showcasing the generalization ability of RL.

### 6.2 Distillation vs. RL for small models

Previous works [DeepSeek-AI et al., 2025] have observed that smaller models relying solely on RL may not be able to achieve performance comparable to those distilled from larger reasoning models. However, our findings contradict this observation: we achieved strong results even with pure RL on

**Table 5:** Cross-domain generalization during math-only and code-only RL for a 24B model

Model	AIME’24	LiveCodeBench v5
Starting Checkpoint	32.2	22.7
RL (Math only)	62.5	38.3 (+15.6)
RL (Code only)	49.7 (+17.5)	42.7



表3: Magistral Small 在不同训练设置下在各个基准测试中的表现。我们报告了三种不同的24B模型的性能：在Magistral Medium的推理轨迹上进行微调的Mistral Small 24B（SFT），从零开始用RL训练的Mistral Small 24B（RL only），以及在Magistral Medium轨迹上进行微调后再用RL增强的Mistral Small 24B（SFT + RL），这是最终的Magistral Small。我们观察到，将推理轨迹微调与RL结合使用可以获得最佳性能。在评估Humanity’s Last Exam时，只考虑了文本子集。

Task	SFT	RL-only	SFT + RL (Magistral Small)
AIME’24 <sub>pass@1</sub>	65.4	65.8	70.7
AIME’24 <sub>maj@64</sub>	90.0	86.7	83.3
AIME’25 <sub>pass@1</sub>	55.6	51.9	62.8
AIME’25 <sub>maj@64</sub>	76.7	66.7	76.7
MATH-500	93.2	95.4	95.9
GPQA	63.4	68.8	68.2
LiveCodeBench (v5)	52.2	46.4	55.8
LiveCodeBench (v6)	44.6	42.4	47.4
Humanity’s Last Exam	5.3	6.1	6.4

表4: Magistral Medium在AIME 2024基准多语言版本上的pass@1表现。

Language	English	French	Spanish	German	Italian	Russian	Chinese
AIME’24 (pass@1)	73.6	68.5	69.3	66.8	66.7	65.0	63.7

## 6 消融实验

在本节中，我们调整训练过程的参数，以研究仅在一种模态上进行强化学习（RL）时会发生什么，比较RL与蒸馏的SFT基线，并阐明我们必须考虑的两个训练选择：批次和小批量大小，以及优势归一化。

### 6.1 跨领域泛化

我们通过在一个领域（数学或代码）上进行训练，然后在另一个领域上进行评估，来研究我们的模型在跨领域泛化方面的能力。具体而言，我们对24B模型进行了两项实验：一项是模型仅在数学数据上训练，并在数学和代码上进行评估，另一项是模型仅在代码上训练，并进行类似的评估。如表5所示，模型在域外任务中表现出色，展示了强化学习的泛化能力。

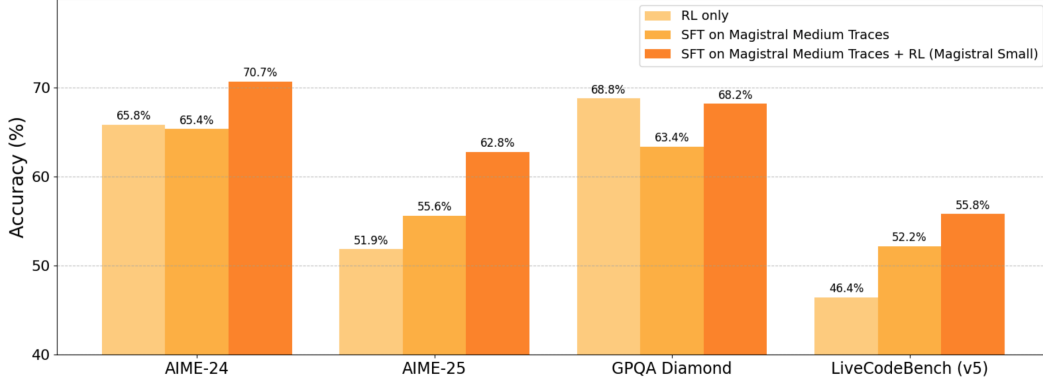
### 6.2 小模型的蒸馏与强化学习（RL）

以往的研究 [DeepSeek-AI 等, 2025] 观察到，仅依赖强化学习（RL）的小模型可能无法达到由更大推理模型蒸馏而来的模型的性能。然而，我们的发现与此相反：即使仅使用纯粹的RL，我们也取得了很好的结果，甚至达到了{v\*}。

表5: 在仅数学和仅代码强化学习中，24B模型的跨领域泛化情况

Model	AIME’24	LiveCodeBench v5
Starting Checkpoint	32.2	22.7
RL (Math only)	62.5	38.3 (+15.6)
RL (Code only)	49.7 (+17.5)	42.7

top of Mistral Small 3. As shown in Figure 5, our Mistral Small 3 with pure RL achieves similar performance on AIME’24 as the distilled version. It even outperforms the distilled version on MATH and GPQA, but has slightly lower performance on code benchmarks such as LiveCodeBench. These results suggest that the benefits of RL are not exclusive to larger base models and hold equally well for smaller models. Furthermore, our findings indicate that the RL on top of the distilled checkpoint can yield even better performance, leading to over 5 points gain across various benchmarks.



**Figure 5: Performance of Magistral Small compared with different training setups on various benchmarks.** We report the performance of three distinct 24B models: Mistral Small 24B trained from scratch with RL (RL only), Mistral Small 24B fine-tuned on reasoning traces from Magistral Medium, and Mistral Small 24B fine-tuned on Magistral Medium traces and subsequently enhanced with RL, which is the final Magistral Small. We observe that the combination of fine-tuning on reasoning traces with RL leads to the best performance.

### 6.3 Batch and minibatch size

Reinforcement learning (RL) algorithms like PPO or GRPO introduce two distinct batch scales. The batch size, denoted as  $n_{\text{batch}}$ , refers to the number of sequences collected before updating the generator’s weights. The minibatch size,  $n_{\text{minibatch}}$ , indicates the number of sequences used to compute the gradient and perform a single optimization step. It is important to note that  $n_{\text{minibatch}}$  must divide  $n_{\text{batch}}$ . Additionally, in an asynchronous RL pipeline, a third scale is introduced: the number of concurrent sequences,  $n_{\text{async}}$ , which represents the number of sequences being generated in parallel. If the number of concurrently generated sequences  $n_{\text{async}}$  is much larger than the batch size  $n_{\text{batch}}$ , a typical sequence was generated with  $n_{\text{async}}/n_{\text{batch}}$  different policies and could be too off-policy. The effect becomes worse as we do more than one minibatch update per one batch.

To test this hypothesis we prepared a strong 3B model using SFT starting from Ministral 3B, and then trained it using GRPO on math-only data with a constant learning rate, a fixed  $n_{\text{async}} = 4096$ , and different values of  $n_{\text{batch}}$  and  $n_{\text{minibatch}}$  in  $\{1024, 2048, 4096, 8192\}$ .

We observe that as long as we keep  $n_{\text{batch}} = n_{\text{minibatch}}$  and that  $n_{\text{batch}}$  is large enough, the performance is very similar when plotted depending on the number of processed prompts, as can be seen in Figure 6 (a). On the other hand, when  $n_{\text{minibatch}}$  is decreased while keeping  $n_{\text{batch}}$  constant, the performance suddenly degrades, even when compared to  $n_{\text{batch}}$  reduced to the same  $n_{\text{minibatch}}$ , as highlighted in Figure 6 (b). When  $n_{\text{batch}} \leq 1024$ , the training becomes less stable, so we opt to keep ratio  $n_{\text{async}}/n_{\text{batch}} \leq 2$  and  $n_{\text{batch}} = n_{\text{minibatch}}$  during final training and further ablations.

### 6.4 Advantage normalization

We experimented with the following advantage normalization methods:

- Minibatch - normalize advantages within a minibatch
- Group normalization - normalize advantages within a group over a single prompt
- No normalization - do not normalize advantages

Previous works [Liu et al., 2025, Andrychowicz et al., 2020] have noted that normalization over a group of generations for a given question can lead to a bias where easy questions or hard questions are

Mistral Small 3的顶部。如图5所示，我们的纯RL的Mistral Small 3在AIME’ 24上的表现与蒸馏版本相似。它甚至在MATH和GPQA上优于蒸馏版本，但在如LiveCodeBench等代码基准测试中的表现略低。这些结果表明，RL的优势并不局限于较大的基础模型，对于较小的模型同样适用。此外，我们的研究还表明，在蒸馏检查点之上进行RL可以获得更好的性能，在各种基准测试中提升超过5点。

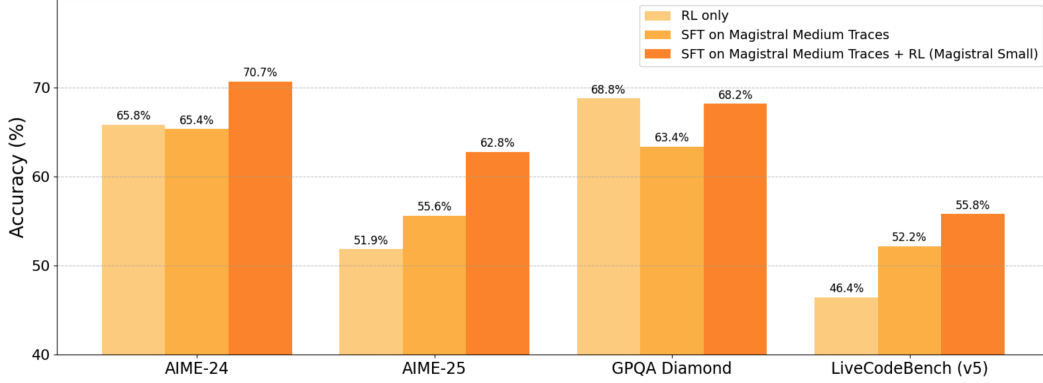


图5: Magistral Small 在不同训练设置下在各种基准测试中的表现。我们报告了三种不同的24B模型的性能：从零开始用RL训练的Mistral Small 24B（仅RL），在Magistral Medium的推理轨迹上微调的Mistral Small 24B，以及在Magistral Medium轨迹上微调后再用RL增强的Mistral Small 24B，即最终的Magistral Small。我们观察到，在推理轨迹上微调结合RL的方法带来了最佳的性能。

### 6.3 批次和小批量大小

强化学习（RL）算法如 PPO 或 GRPO 引入了两种不同的批次规模。批次大小，记为  $n_{\text{batch}}$ ，指在更新生成器权重之前收集的序列数量。小批量大小， $n_{\text{minibatch}}$ ，表示用于计算梯度和执行单次优化步骤的序列数量。需要注意的是， $n_{\text{minibatch}}$  必须整除  $n_{\text{batch}}$ 。此外，在异步 RL 流水线中，还引入了第三个尺度：并发序列数， $n_{\text{async}}$ ，代表同时生成的序列数量。如果并发生成的序列数  $n_{\text{async}}$  远大于批次大小  $n_{\text{batch}}$ ，则一个典型的序列是用  $n_{\text{async}}/n_{\text{batch}}$  种不同的策略生成的，可能会偏离策略太远。随着每个批次进行多于一次的小批量更新，这个影响会变得更糟。

为了验证这个假设，我们使用从 Ministral 3B 开始的 SFT 训练了一个强大的 3B 模型，然后在仅包含数学数据的情况下，采用恒定学习率，固定  $n_{\text{async}} = 4096$ ，以及不同的  $n_{\text{batch}}$  和  $n_{\text{minibatch}}$  在  $\{1024, 2048, 4096, 8192\}$  的值，使用 GRPO 进行训练。

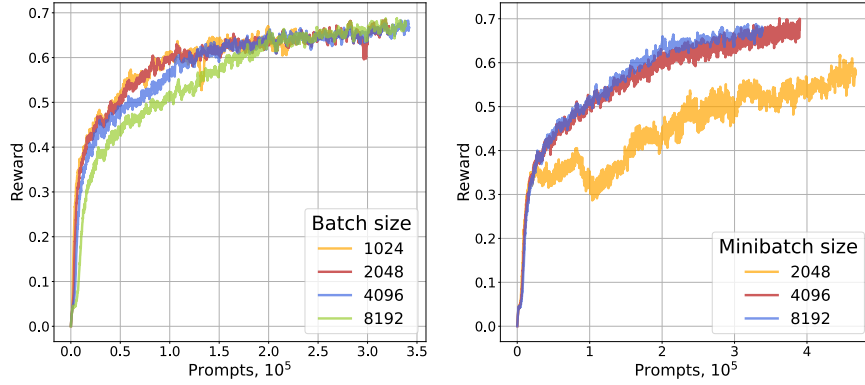
我们观察到，只要保持  $n_{\text{batch}} = n_{\text{minibatch}}$  并且  $n_{\text{batch}}$  足够大，性能在根据处理的提示数量绘制时非常相似，如图6（a）所示。另一方面，当  $n_{\text{minibatch}}$  减小时，同时保持  $n_{\text{batch}}$  不变，性能会突然下降，即使与将  $n_{\text{batch}}$  减少到相同的  $n_{\text{minibatch}}$  时相比也是如此，如图6（b）所强调。当  $n_{\text{batch}} \leq 1024$  时，训练变得不那么稳定，因此我们选择在最终训练和进一步消融实验中保持比例  $n_{\text{async}}/n_{\text{batch}} \leq 2$ ，以及保持  $n_{\text{batch}} = n_{\text{minibatch}}$ 。

### 6.4 优势归一化

我们尝试了以下优势归一化方法：

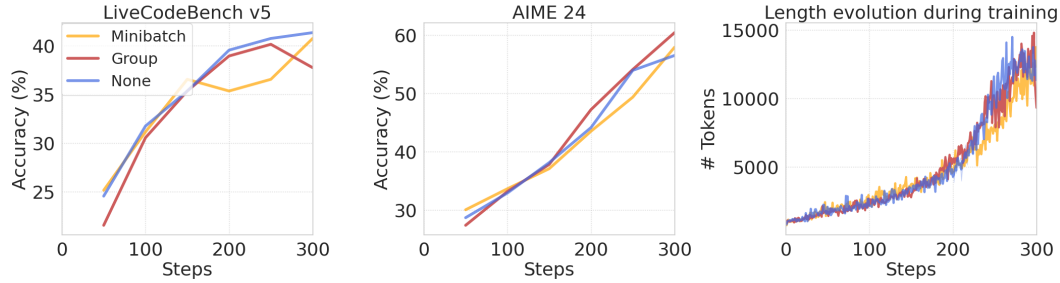
- 小批量 - 在小批量内归一化优势
- 组归一化 - 在单个提示中对组内的优势进行归一化
- 无归一化 - 不要归一化优势

之前的研究 [Liu et al., 2025, Andrychowicz et al., 2020] 指出，对给定问题的一组生成结果进行归一化可能会导致偏差，使得容易的问题或困难的问题被偏向。



**Figure 6: Impact of batch and minibatch sizes on RL training rewards.** (a) Reward during RL training of 3B model on math data for different batch sizes, while keeping minibatch size equal to batch size. Number of concurrently generated sequences is kept constant at 4096. (b) Reward during RL training in the same setup for different minibatch sizes at fixed batch size of 8192 sequences. We observe that performance doesn’t depend strongly on batch size, but degrades when there are more than 2 minibatches in a batch.

upweighted due to their lower standard deviation values. However, we did not observe any significant effects on evaluation performance or the growth of the length as shown in Figure 7. Hence, we decided to use minibatch normalization for all our experiments.



**Figure 7: Results for training with different advantage normalizations in GRPO.** We observe that different normalization methods do not lead to significant difference either in evaluation performance or the length growth during training.

## 7 Analysis

In this section, we investigate the dynamics of RL training and present evidence that increasing completion length is the main resource that improves the performance of the model. Those dynamics are not destructive to previous capabilities, and the reasoning capabilities can even generalize: multimodal reasoning gets improved for free, and function calling and instruction following remain unchanged or even get a small boost. Additionally, we discuss two ideas that didn’t work for us - giving more fine-grained rewards in code tasks based on test completion rate and controlling entropy via entropy bonus term in the loss.

### 7.1 Reinforcement learning moves weights in low-dimensional space

To better understand the dynamics of Magistral during RL training, we follow the method of [Li et al., 2018] to analyze the **Magistral Small RL-only** run and visualize the loss landscape around the final checkpoint.

First, we stack the weights of all intermediate checkpoints in a matrix  $X \in \mathbb{R}^{T \times W}$ , where  $T$  is the number of checkpoints and  $W$  is the number of weights. Then, we subtract the mean weights

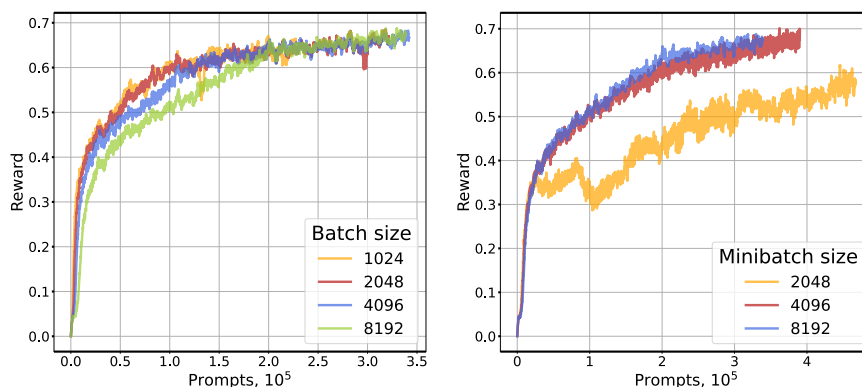


图6: 批次和小批量大小对强化学习训练奖励的影响。(a) 在数学数据上对3B模型进行强化学习训练时，不同批次大小的奖励，同时保持小批量大小等于批次大小。并发生成序列的数量保持在4096不变。(b) 在相同设置下，固定批次大小为8192序列，不同小批量大小的强化学习训练奖励。我们观察到，性能并不强烈依赖于批次大小，但当批次中有超过2个小批量时，性能会下降。

由于它们的标准差值较低而被赋予更高的权重。然而，我们在评估性能或长度增长方面未观察到任何显著影响，如图7所示。因此，我们决定在所有实验中使用小批量归一化。

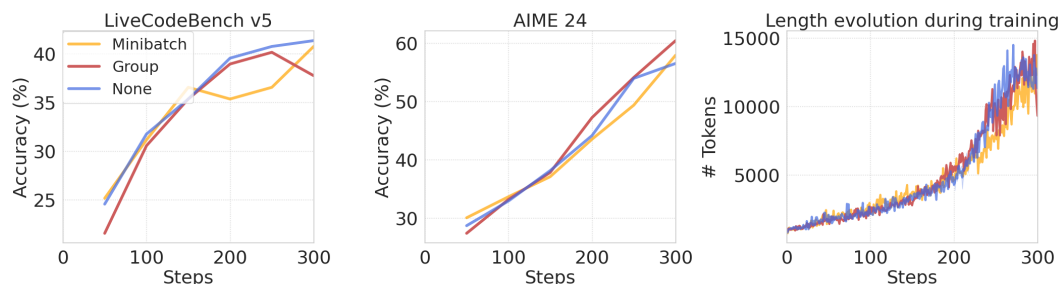


图7: 在GRPO中使用不同优势归一化的训练结果。我们观察到，不同的归一化方法在评估性能或训练过程中的长度增长方面没有显著差异。

## 7 分析

在本节中，我们研究了强化学习训练的动态，并提供证据表明，增加完成长度是提升模型性能的主要资源。这些动态对之前的能力没有破坏，甚至推理能力还能得到泛化：多模态推理得到了免费提升，函数调用和指令遵循保持不变，甚至还略有提升。此外，我们还讨论了两两种未能奏效的想法——在代码任务中根据测试完成率给予更细粒度的奖励，以及通过在损失中加入熵奖励项来控制熵。

### 7.1 强化学习在低维空间中移动权重

为了更好地理解Magistral在RL训练中的动态，我们采用[Li等人，2018]的方法，分析Magistral的仅RL运行，并可视化最终检查点周围的损失景观。

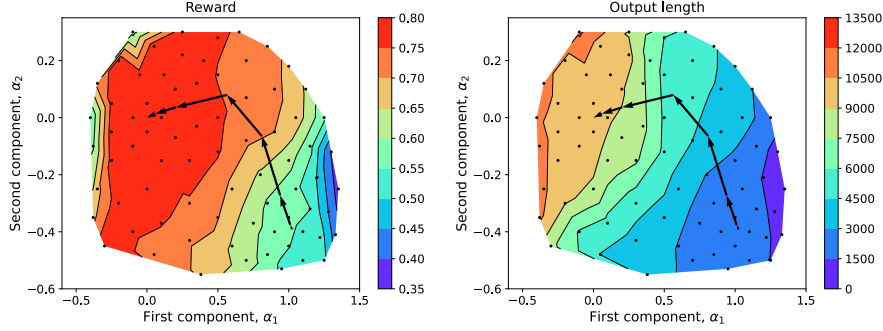
首先，我们将所有中间检查点的权重堆叠成一个矩阵  $X \in \mathbb{R}^{T \times W}$ ，其中  $T$  是检查点的数量， $W$  是权重的数量。然后，我们减去平均权重

across the  $T$  checkpoints and perform a PCA analysis to find two principal components of the matrix  $X$  in the weight space. Since the weight space is very high-dimensional, we use the iterative Lanczos-Arnoldi algorithm [Saad, 2003] to find the top-2 eigenvectors of  $X^T X$ . As a result, we obtain two components  $c_1$  and  $c_2$  that we L2-normalize to have a unit norm.

Second, we perturb the final checkpoint weights  $w^* \in \mathbb{R}^W$  by adding two components as

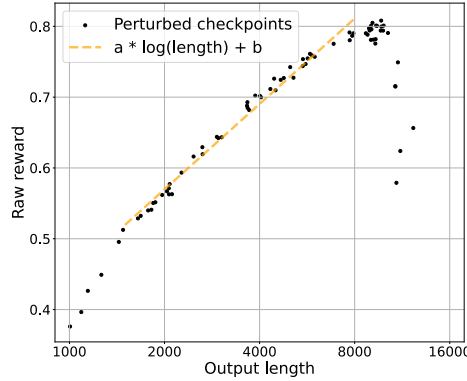
$$w(\alpha_1, \alpha_2) = w^* + \alpha_1 c_1 + \alpha_2 c_2 \quad (2)$$

We evaluate each perturbed checkpoint on a fixed batch of 512 prompts, generating 16 completions per prompt, and using the same reward setting as in **Magistral Small RL-only** run. Finally, we compute mean reward and mean output length for each checkpoint and plot it in  $(\alpha_1, \alpha_2)$  coordinates.



**Figure 8: Reward and length evolution in  $w(\alpha_1, \alpha_2)$  hyperplane.** Black arrow trajectory is a projection of intermediate checkpoints of **Magistral Small RL-only** run on the hyperplane. Black points are perturbed checkpoints computed using Equation 2. Intermediate values are computed with linear interpolation on the triangular grid.

We clearly observe that there is a “length” direction - as model goes from right to left in Figure 8, mean reward and output length grow up until the point where length starts to hit length penalty and maximally allowed completion length. We additionally plot dependence of raw reward without length penalty on output length, observing a ubiquitous log scaling in Figure 9.



**Figure 9: Reward scaling with output length.** Each point corresponds to a perturbed checkpoint computed with Equation 2. We generate 8192 completions with the checkpoint and evaluate mean output length and raw reward (reward without length penalty). We perform linear regression on checkpoints with mean output length between 1500 and 8000 and observe that reward scales logarithmically with the output length.

## 7.2 Eating the multimodal free lunch

The initial checkpoints utilized for RL training, Mistral Small 3 and Mistral Medium 3, are multimodal models and come with associated vision encoders. During the RL training phase, as the models are



穿过  $T$  检查点并进行 PCA 分析，以在权重空间中找到矩阵  $X$  的两个主成分。由于权重空间维度非常高，我们使用迭代的 Lanczos-Arnoldi 算法 [Saad, 2003] 来找到  $X^T X$  的前两个特征向量。结果，我们得到两个分量  $c_1$  和  $c_2$ ，并对它们进行 L2 归一化，使其范数为1。

其次，我们通过添加两个分量来扰动最终检查点权重  $w^* \in \mathbb{R}^W$ 。

$$w(\alpha_1, \alpha_2) = w^* + \alpha_1 c_1 + \alpha_2 c_2 \quad (2)$$

我们在一组固定的512个提示上评估每个扰动的检查点，每个提示生成16个完成，并使用与 Magistral Small RL-only 运行相同的奖励设置。最后，我们计算每个检查点的平均奖励和平均输出长度，并在  $(\alpha_1, \alpha_2)$  坐标中绘制。

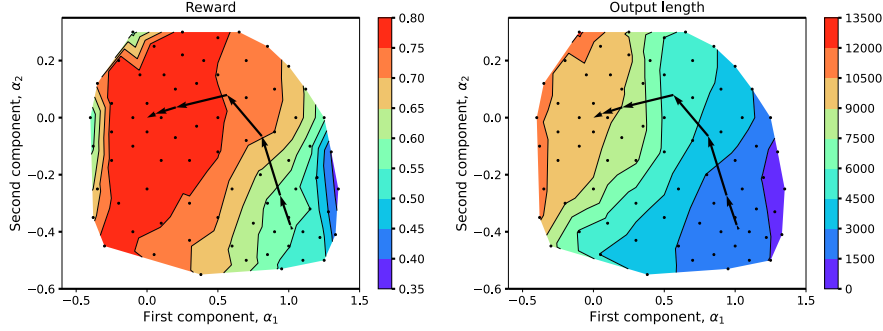


图8:  $w(\alpha_1, \alpha_2)$  超平面中的奖励和长度演变。黑色箭头轨迹是Magistral Small仅RL运行在超平面上的中间检查点的投影。黑色点是使用方程2计算的扰动检查点。中间值通过在三角网格上进行线性插值计算。

我们清楚地观察到存在一个“长度”方向——如图8中模型从右向左移动时，平均奖励和输出长度不断增长，直到长度开始受到长度惩罚和最大允许完成长度的限制。我们还绘制了未加长度惩罚的原始奖励与输出长度的关系，观察到在图9中普遍存在对数尺度的变化。

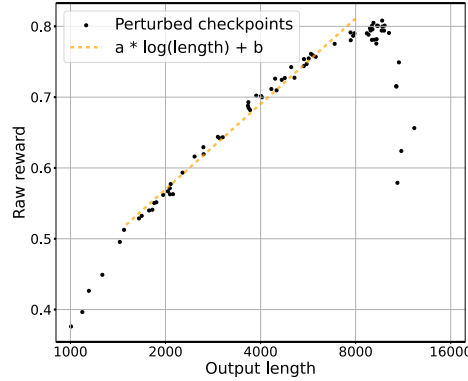
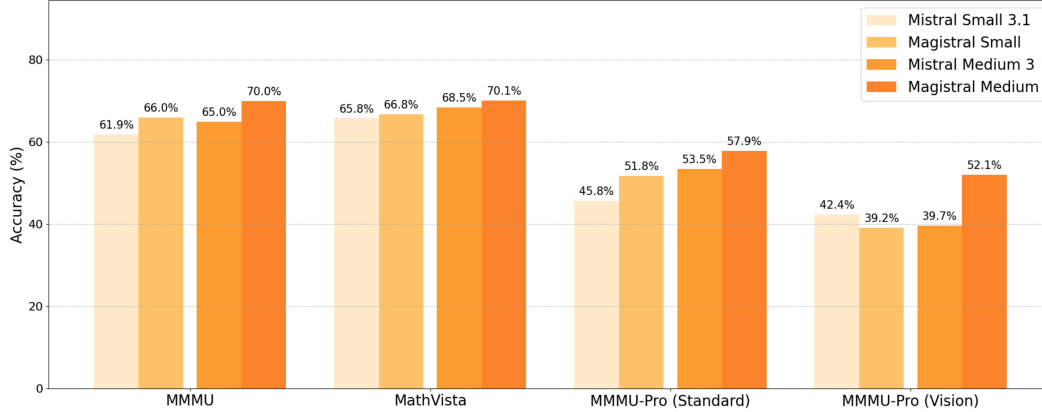


图9: 奖励与输出长度的缩放关系。每个点对应用方程 2 计算的扰动检查点。我们用该检查点生成 81 92 个完成，并评估平均输出长度和原始奖励（不含长度惩罚的奖励）。我们对平均输出长度在 1500 到 8000 之间的检查点进行线性回归，观察到奖励与输出长度呈对数关系缩放。

## 7.2 吃多模态免费午餐

用于强化学习训练的初始检查点，Mistral Small 3 和 Mistral Medium 3，是多模态模型，并配备了相关的视觉编码器。在强化学习训练阶段，由于模型是



**Figure 10: Performance on multimodal benchmarks.**

trained on text-only data, one might expect the multimodal performance to degrade. However, on the contrary, we discover that the models not only retain their multimodal capabilities, but unexpectedly develop enhanced multimodal reasoning abilities. The resulting models also showcase improved performance on multimodal benchmarks.

We report results multimodal benchmarks designed to assess reasoning capabilities, specifically MathVista [Lu et al., 2024], MMMU [Yue et al., 2024], and MMMU-Pro [Yue et al., 2025]. Our results in Figure 10 show no performance regression across most benchmarks, with notable improvements observed on MMMU (+5%, reaching 70%), MMMU-Pro-Standard (+4.4%, reaching 57.9%) and MMMU-Pro-Vision (+12%, reaching 52.1%). While the most significant improvements are seen in scientific questions that require textual reasoning, we observe that the model transfers its extended thinking process across all types of questions (see figures 14, 15, 16 for qualitative examples).

### 7.3 Impact of RL on other capabilities

Similar to the multimodal capabilities mentioned in Section 7.2, our RL checkpoint maintains and even improves its tool calling and instruction following capabilities [Zhou et al., 2023] (Table 6). This allows us to integrate the model out-of-the-box with existing tools as shown.

**Table 6: Benchmarks before and after reinforcement learning.** Internal bench is Mistral’s internal function calling benchmark. We use an internal version of IFEval that fixes some issues with the public version. The scores are not comparable with other publicly shared scores.

Category	Benchmark	Mistral Medium 3	Magistral Medium
Function calling	Internal bench	87.2	87.4
Instruction following	IFEval	86.8	87.4

### 7.4 Unsuccessful approaches

In this section, we present various approaches that we tried but ultimately did not adopt, as they did not yield any performance improvements.

#### 7.4.1 Partial reward for code data

The strict requirements of competitive programming, in terms of correctness and adherence to complexity constraints, result in sparse rewards, often causing many code generations to be discarded due to limited reward diversity.

To address this, we experimented with a proportional reward: based on the fraction of tests passed, as opposed to the binary reward discussed in Section 2.2.2. In an ablation with a 24B model over 250

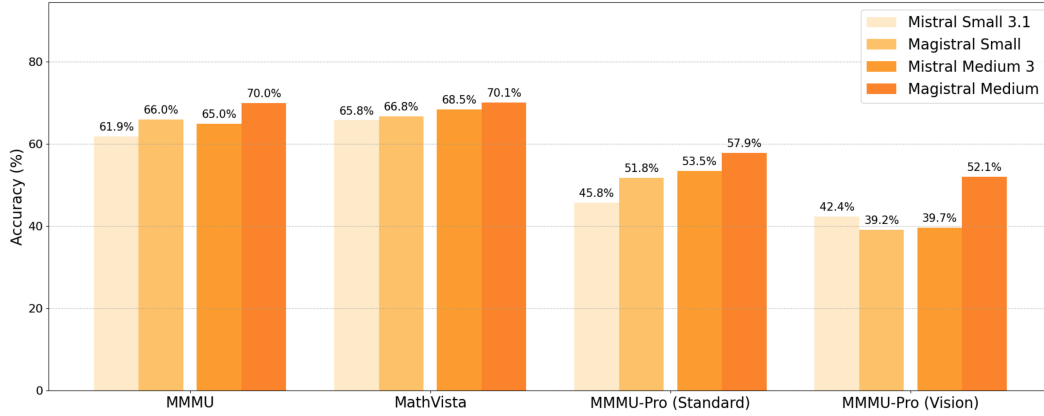


图10: 多模态基准测试的性能。

仅在纯文本数据上进行训练，可能会预期多模态性能会下降。然而，恰恰相反，我们发现这些模型不仅保持了它们的多模态能力，而且意外地发展出了增强的多模态推理能力。由此产生的模型在多模态基准测试中的表现也得到了提升。

我们报告了旨在评估推理能力的多模态基准测试结果，具体包括 MathVista [Lu 等, 2024]、MMMU [Yue 等, 2024] 和 MMMU-Pro [Yue 等, 2025]。我们在图10中的结果显示，在大多数基准测试中没有性能回退，在 MMMU (+5%，达到70%)、MMMU-Pro-Standard (+4.4%，达到57.9%) 和 MMMU-Pro-Vision (+12%，达到52.1%) 上观察到显著的提升。虽然在需要文本推理的科学问题中观察到最显著的改进，但我们也发现模型将其扩展的思维过程迁移到所有类型的问题中（请参见图14、15、16的定性示例）。

### 7.3 强化学习对其他能力的影响

与第7.2节提到的多模态能力类似，我们的RL检查点保持甚至提升了其工具调用和指令执行能力 [Zhou et al., 2023]（表6）。这使我们能够如图所示，将模型开箱即用集成到现有工具中。

表6: 强化学习前后的基准测试。内部基准是Mistral的内部函数调用基准测试。我们使用了修正了部分公共版本问题的内部版本IFEval。分数无法与其他公开分享的分数进行比较。

Category	Benchmark	Mistral Medium 3	Magistral Medium
Function calling	Internal bench	87.2	87.4
Instruction following	IFEval	86.8	87.4

### 7.4 失败的方法

在本节中，我们介绍了我们尝试过但最终未采用的各种方法，因为它们未能带来任何性能提升。

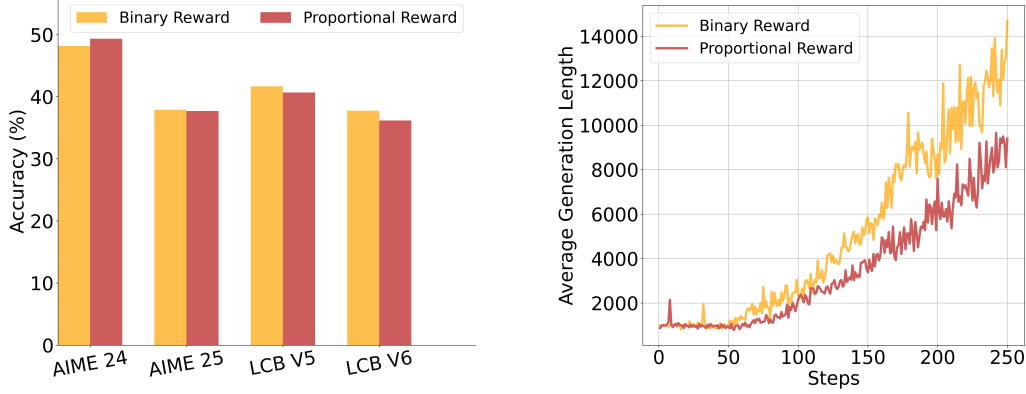
#### 7.4.1 代码数据的部分奖励

竞争编程在正确性和复杂度约束方面的严格要求导致奖励稀疏，常常因为奖励多样性有限而使许多代码生成被舍弃。

为了解决这个问题，我们尝试了一种比例奖励：基于通过测试的比例，而不是第2.2.2节中讨论的二元奖励。在一个使用24B模型进行的消融实验中，进行了250次试验。

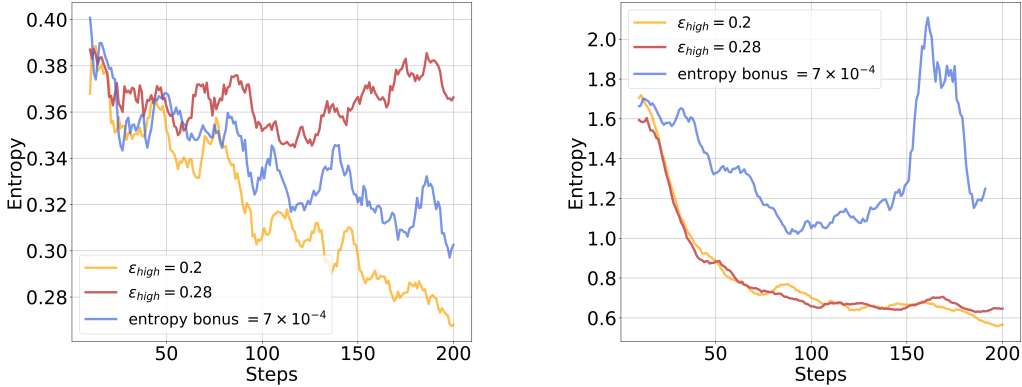
steps, we found that training with proportional rewards was faster, discarding three times less data. However, this approach led to slightly lower final performance on benchmarks, with a 2% decrease on LiveCodeBench (Figure 11a), and slower growth in generation length (Figure 11b).

The hope was that a reward based on the fraction of tests passed should provide a richer signal than a simple pass/fail for RL training. However, the potential issue is that partial rewards could also provide false signal to incorrect solutions and be more sensitive to minor inconsistencies between implementations, potentially leading to less meaningful training batches.



**Figure 11: Binary vs proportional reward for code problems.** (a) Accuracy on AIME and LiveCodeBench after 250 steps of training with binary reward and proportional reward. Performance on LiveCodeBench is 2% lower with proportional rewards. (b) Length evolution throughout training. Length increases more with binary rewards.

#### 7.4.2 Entropy targeting



**Figure 12: Impact of  $\epsilon_{high}$  on the entropy distribution throughout training.** (a) Entropy evolution throughout training of a 3B model on a **math only** dataset. Entropy drops with entropy bonus, while higher  $\epsilon_{high}$  maintains entropy, allowing for better exploration. (b) Entropy evolution throughout training of a 3B model on a **math and code** dataset. Entropy explodes with entropy bonus, even though the coefficient is the same as the math only version. Higher  $\epsilon_{high}$  behaves better, allowing entropy to decrease.

To encourage exploration and prevent entropy collapse during RL training, a common strategy in the literature [Schulman et al., 2017] is to add an entropy bonus loss term. However, we found this strategy to be unstable as the effect of the entropy bonus varies significantly depending on the dataset. For a math-only dataset, entropy drops with the entropy bonus, while a higher  $\epsilon_{high}$  maintains entropy, enhancing exploration (Figure 12a). On a math and code dataset, entropy increases excessively with the entropy bonus (even with the same coefficient as in the math-only run), while a higher  $\epsilon_{high}$  allows entropy to decrease, improving exploitation (Figure 12b).

步骤中，我们发现使用比例奖励进行训练更快，丢弃的数据也少了三倍。然而，这种方法在基准测试中的最终性能略低，LiveCodeBench上的表现下降了2%（图11a），且生成长度的增长速度也较慢（图11b）。

希望是基于通过的测试比例的奖励应该比简单的通过/不通过提供更丰富的信号，用于强化学习训练。然而，潜在的问题是部分奖励也可能向错误的解决方案提供虚假信号，并且对实现之间的细微不一致更敏感，可能导致训练批次的意义降低。

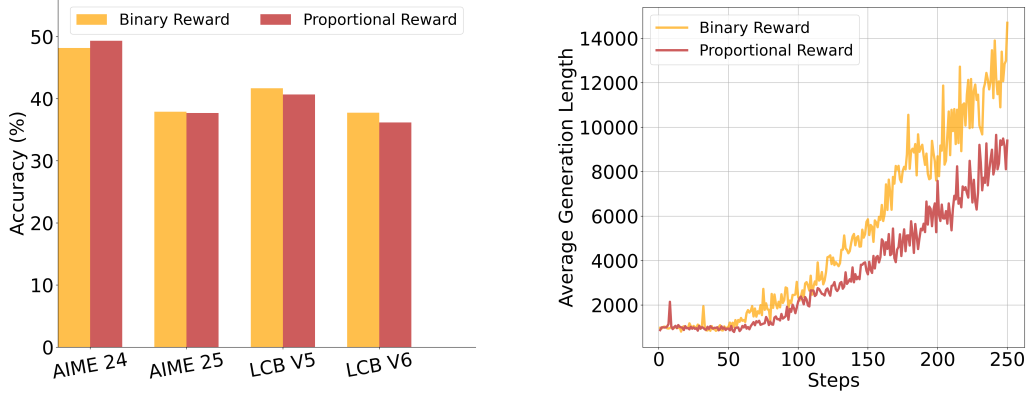


图11：二元奖励与比例奖励在代码问题上的比较。(a) 在经过250步训练后，二元奖励和比例奖励在AIME和LiveCodeBench上的准确率。使用比例奖励时，LiveCodeBench的性能低2%。(b) 训练过程中长度的变化。使用二元奖励时，长度增长更快。

#### 7.4.2 熵目标

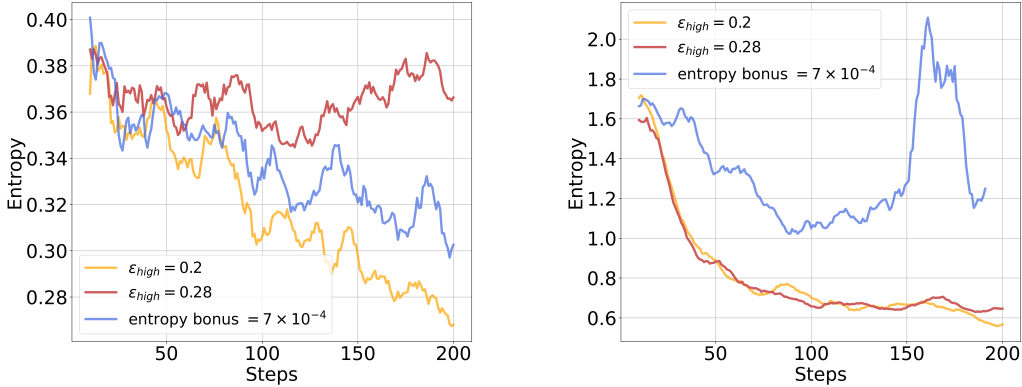
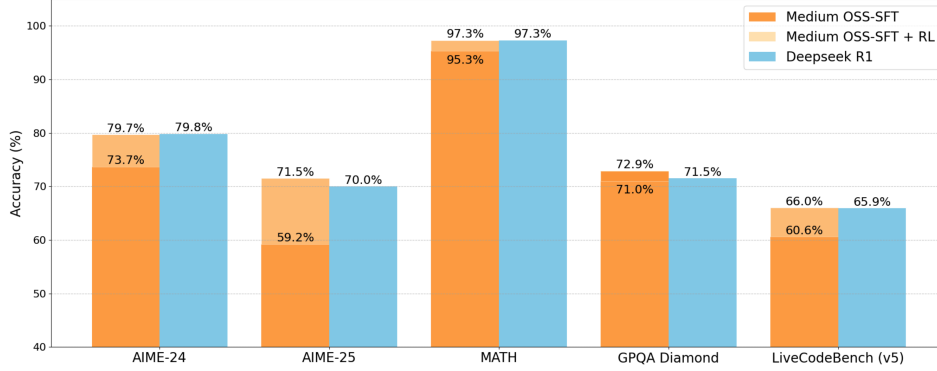


图12： $\epsilon_{high}$  对训练过程中熵分布的影响。(a) 在仅包含数学数据集上训练一个3B模型的熵演变。加入熵奖励后，熵下降，而较高的 $\epsilon_{high}$ 维持熵，从而实现更好的探索。(b) 在包含数学和代码数据集上训练一个3B模型的熵演变。尽管系数与仅数学版本相同，加入熵奖励后熵急剧上升。较高的 $\epsilon_{high}$ 表现更佳，允许熵下降。

为了鼓励探索并防止在强化学习训练过程中熵的崩溃，文献中常用的策略[Schulman et al., 2017]是加入一个熵奖励损失项。然而，我们发现这一策略不稳定，因为熵奖励的效果在很大程度上取决于数据集的不同。在仅包含数学数据集上，熵随着熵奖励的增加而下降，而较高的 $\epsilon_{high}$ 则保持了熵，增强了探索（图12a）。在包含数学和代码的数据集上，熵随着熵奖励的增加而过度上升（即使系数与仅数学数据集相同），而较高的 $\epsilon_{high}$ 则允许熵下降，从而改善了利用（图12b）。

Instead, we found it more effective to depend on  $\varepsilon_{\text{high}}$ , as also noted in literature [Yu et al., 2025, Wang et al., 2025]. This method avoids the instability issues associated with entropy bonuses.

Another approach for controlling entropy is adding a KL term to the PPO loss. However, as the generation distribution is expected to deviate significantly from the original model, we found that using a KL penalty primarily hinders training, consistent with previous findings [Yu et al., 2025]. We attempted using an exponential moving average of the weights during training as a reference for KL, but found it simpler to manually adjust  $\varepsilon_{\text{high}}$ .



**Figure 13: Benchmark performance of Magistral Medium fine-tuned on open-source traces.** All results are reported using pass@1. The shaded region highlights the additional improvement achieved through RL on top of supervised fine-tuning. We find that while fine-tuning on open-source traces yields strong results, applying RL further enhances performance significantly. In particular, the accuracy on AIME’25 increases by more than 12%. Please note that the performance on GPQA Diamond drops after RL from 72.9% to 71.0%.

## 8 RL on model finetuned using OSS reasoning traces

As an experiment, we also tried to first finetune Mistral Medium 3 using open source reasoning datasets OpenThoughts [Guha et al., 2025] and the code subset of OpenR1 [Hugging Face, 2025, Penedo et al., 2025] including both the prompts and the generations from these datasets i.e. Deepseek R1 generated traces. This included a total of about 1.3M generations. We then run RL on top of this finetuned checkpoint using our most difficult subset of the data. As shown in Figure 13, applying RL yields substantial performance gains over the SFT checkpoint. Notably, the RL model improves by over 10 points on AIME’25 and 5 points on LiveCodeBench, achieving a final performance level on par with Deepseek-R1 on code and math benchmarks.

## 9 Conclusion

Magistral is our first step towards generally capable systems with reinforcement learning. We look forward to the next research problems ahead of us: what loss and optimization algorithms are the most appropriate, how much gain can be unlocked by bootstrapping a model on its own reasoning traces, or how to scale to the next order of magnitude of compute. Looking ahead, we are also excited to push the boundaries of RL across a whole range of applications, with tool-use, integrated multimodality, and agents. As we explore this frontier, we remain committed to contributing to science in a transparent and optimistic manner.



相反，我们发现依赖于  $\epsilon_{\text{high}}$  更为有效，正如文献 [Yu et al., 2025, Wang et al., 2025] 中所指出的。这种方法避免了与熵奖励相关的不稳定问题。

另一种控制熵的方法是在PPO损失中加入KL项。然而，由于生成分布预计会显著偏离原始模型，我们发现使用KL惩罚主要会阻碍训练，这与之前的研究结果[Yu et al., 2025]一致。我们尝试在训练过程中使用权重的指数移动平均作为KL的参考，但发现手动调整 $\epsilon_{\text{high}}$ 更为简单。

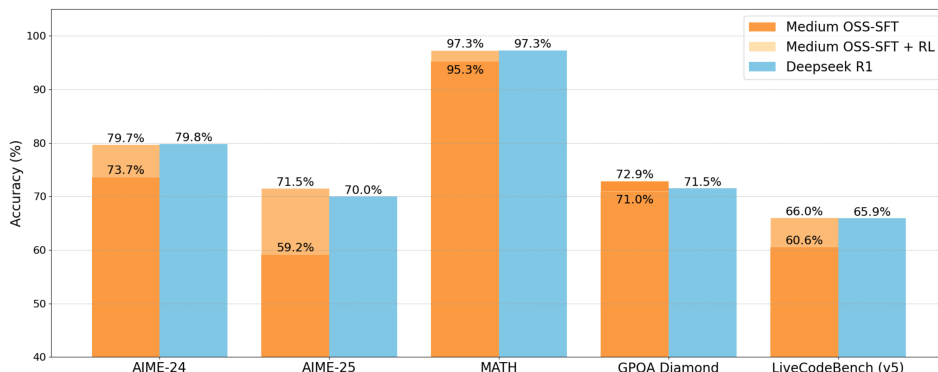


图13: Magistral Medium在开源轨迹上微调的基准性能。所有结果均以pass@1报告。阴影区域突出显示了在监督微调基础上通过RL实现的额外提升。我们发现，虽然在开源轨迹上微调可以获得较强的结果，但应用RL可以进一步显著提升性能。特别是，AIME’ 25的准确率提高了超过12%。请注意，经过RL后，GPQA Diamond的性能从72.9%下降到71.0%。

## 使用OSS推理轨迹微调模型的8个RL

作为一项实验，我们还尝试先使用开源推理数据集OpenThoughts [Guha等, 2025]和OpenR1 [Hugging Face, 2025, Penedo等, 2025]的代码子集对Mistral Medium 3进行微调，包括这些数据集中的提示和生成内容，即Deepseek R1生成的轨迹。这包括大约130万次生成。然后，我们在这个微调的检查点上使用我们最困难的数据子集进行强化学习（RL）。如图13所示，应用RL在性能上明显优于SFT检查点。值得注意的是，RL模型在AIME’ 25上提升了超过10分，在LiveCodeBench上提升了5分，最终在代码和数学基准测试中达到了与Deepseek-R1相当的性能水平。

## 9 结论

Magistral 是我们迈向具有强化学习能力的通用系统的第一步。我们期待着未来的研究课题：哪些损失函数和优化算法最为合适，如何通过模型自身推理轨迹上引导，解锁更多的收益，或者如何扩展到下一数量级的计算能力。展望未来，我们也非常期待在一系列应用中推动强化学习的边界，包括工具使用、集成多模态以及智能体。在探索这一前沿的过程中，我们将继续以透明和乐观的态度，为科学做出贡献。

### **Core contributors**

Abhinav Rastogi, Albert Q. Jiang, Andy Lo, Gabrielle Berrada, Guillaume Lample, Jason Rute, Joep Barmmentlo, Karmesh Yadav, Kartik Khandelwal, Khyathi Raghavi Chandu, Léonard Blier, Lucile Saulnier, Matthieu Dinot, Maxime Darrin, Neha Gupta, Roman Soletskyi, Sagar Vaze, Teven Le Scao, Yihan Wang

### **Contributors**

Adam Yang, Alexander Liu, Alexandre Sablayrolles, Amélie Héliou, Amélie Martin, Andy Ehrenberg, Anmol Agarwal, Antoine Roux, Arthur Darcet, Arthur Mensch, Baptiste Bout, Baptiste Rozière, Baudouin De Monicault, Chris Bamford, Christian Wallenwein, Christophe Renaudin, Clémence Lanfranchi, Darius Dabert, Devon Mizelle, Diego de las Casas, Elliot Chane-Sane, Emilien Fugier, Emma Bou Hanna, Gauthier Delerce, Gauthier Guinet, Georgii Novikov, Guillaume Martin, Himanshu Jaju, Jan Ludziejewski, Jean-Hadrien Chabran, Jean-Malo Delignon, Joachim Studnia, Jonas Amar, Josselin Somerville Roberts, Julien Denize, Karan Saxena, Kush Jain, Lingxiao Zhao, Louis Martin, Luyu Gao, Léo Renard Lavaud, Marie Pellat, Mathilde Guillaumin, Mathis Felardos, Maximilian Augustin, Mickaël Seznec, Nikhil Raghuraman, Olivier Duchenne, Patricia Wang, Patrick von Platen, Patryk Saffer, Paul Jacob, Paul Wambergue, Paula Kurylowicz, Pavankumar Reddy Muddireddy, Philomène Chagniot, Pierre Stock, Pravesh Agrawal, Romain Sauvestre, Rémi Delacourt, Sanchit Gandhi, Sandeep Subramanian, Shashwat Dalal, Siddharth Gandhi, Soham Ghosh, Srijan Mishra, Sumukh Aithal, Szymon Antoniak, Thibault Schueller, Thibaut Lavril, Thomas Robert, Thomas Wang, Timothée Lacroix, Valeriia Nemychnikova, Victor Paltz, Virgile Richard, Wen Ding Li, William Marshall, Xuanyu Zhang, Yunhao Tang

## 核心贡献者

阿比纳夫·拉斯托吉, 艾伯特·Q·江, 安迪·洛, 加布里埃尔·贝拉达, 纪尧姆·拉姆普尔, 杰森·鲁特, 约普·巴门特洛, 卡梅什·雅达夫, 卡尔蒂克·坎德尔瓦尔, 卡雅蒂·拉加维·钱杜, 莱昂纳尔·布利耶, 吕西尔·索尔尼耶, 马修·迪诺, 马克西姆·达林, 内哈·古普塔, 罗曼·索莱茨基, 萨加尔·瓦泽, 特文·勒斯考, 王一涵

## 贡献者

亚当·杨, 亚历山大·刘, 亚历山大·萨布莱洛斯, 阿梅莉·赫利乌, 阿梅莉·马丁, 安迪·埃伦伯格, 安莫尔·阿加瓦尔, 安托万·鲁, 阿瑟·达尔塞, 阿瑟·门施, 巴蒂斯特·布特, 巴蒂斯特·罗齐耶, 鲍丁·德·莫尼考, 克里斯·班福德, 克里斯蒂安·沃伦韦因, 克里斯托弗·雷诺丹, 克莱芒斯·兰弗兰奇, 达里乌斯·达贝尔特, 德文·米泽尔, 迭戈·德拉斯卡萨斯, 埃利奥特·查恩·萨恩, 埃米利安·弗吉耶, 艾玛·布哈纳, 戈蒂耶·德莱尔斯, 戈蒂耶·吉内, 乔治·诺维科夫, 纪尧姆·马丁, 希曼舒·贾朱, 扬·卢德泽列夫斯基, 让·哈德里安·查布兰, 让-马洛·德利农, 约阿希姆·斯图迪亚, 乔纳斯·阿马尔, 乔斯林·索默维尔·罗伯茨, 朱利安·德尼兹, 卡兰·萨克塞纳, 库什·Jain, 赵凌霄, 路易斯·马丁, 高鲁玉, 雷尼尔·雷纳尔·拉沃, 玛丽·佩拉特, 马蒂尔德·吉洛曼, 马蒂斯·费拉多斯, 马克西米连·奥古斯丁, 米卡埃尔·塞泽克, 尼克希尔·拉古拉曼, 奥利维耶·杜申, 帕特里夏·王, 帕特里克·冯·普拉滕, 帕特里克·萨弗, 保罗·雅各布, 保罗·瓦姆贝尔格, 宝拉·库尔伊沃维奇, 帕万库马尔·雷迪·穆迪雷迪, 菲洛梅娜·查尼奥, 皮埃尔·斯托克, 普拉韦什·阿格拉瓦尔, 罗曼·索韦斯特, 雷米·德拉库尔, 桑奇特·甘地, 桑迪普·苏布拉马尼安, 沙什瓦特·达拉尔, 西达尔特·甘地, 索哈姆·戈什, 斯里詹·米什拉, 苏穆克·艾塔尔, 席莫恩·安东尼亚克, 蒂博·舒勒, 蒂博·拉弗里尔, 托马斯·罗伯特, 托马斯·王, 蒂莫泰·拉克鲁瓦, 瓦莱里娅·涅米钦科娃, 维克多·帕尔茨, 维尔吉尔·理查德, 丁威廉, 威廉·马歇尔, 张轩宇, 唐云浩

## References

- Marcin Andrychowicz, Anton Raichuk, Piotr Stańczyk, Manu Orsini, Sertan Girgin, Raphael Marinier, Léonard Hussenot, Matthieu Geist, Olivier Pietquin, Marcin Michalski, Sylvain Gelly, and Olivier Bachem. What matters in on-policy reinforcement learning? a large-scale empirical study, 2020. URL <https://arxiv.org/abs/2006.05990>
- DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojuan Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-rl: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL <https://arxiv.org/abs/2501.12948>
- Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Vlad Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, et al. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. In *International conference on machine learning*, pages 1407–1416. PMLR, 2018.
- Paul Gauthier. Polyglot Benchmark. <https://github.com/Aider-AI/polyglot-benchmark>, 2024. URL <https://github.com/Aider-AI/polyglot-benchmark>. GitHub repository. Coding problems sourced from Exercism language tracks.
- Etash Guha, Ryan Marten, Sedrick Keh, Negin Raoof, Georgios Smyrnis, Hritik Bansal, Marianna Nezhurina, Jean Mercat, Trung Vu, Zayne Sprague, Ashima Suvarna, Benjamin Feuer, Liangyu Chen, Zaid Khan, Eric Frankel, Sachin Grover, Caroline Choi, Niklas Muennighoff, Shiye Su, Wanjia Zhao, John Yang, Shreyas Pimpalgaonkar, Kartik Sharma, Charlie Cheng-Jie Ji, Yichuan Deng, Sarah Pratt, Vivek Ramanujan, Jon Saad-Falcon, Jeffrey Li, Achal Dave, Alon Albalak, Kushal Arora, Blake Wulfe, Chinmay Hegde, Greg Durrett, Sewoong Oh, Mohit Bansal, Saadia Gabriel, Aditya Grover, Kai-Wei Chang, Vaishaal Shankar, Aaron Gokaslan, Mike A. Merrill, Tatsunori Hashimoto, Yejin Choi, Jenia Jitsev, Reinhard Heckel, Maheswaran Sathiamoorthy, Alexandros G. Dimakis, and Ludwig Schmidt. Openthoughts: Data recipes for reasoning models, 2025. URL <https://arxiv.org/abs/2506.04178>
- Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

Marcin Andrychowicz, Anton Raichuk, Piotr Stańczyk, Manu Orsini, Sertan Girgin, Raphael Marinier, Léonard Hussenot, Matthieu Geist, Olivier Pietquin, Marcin Michalski, Sylvain Gelly 和 Olivier Bachem。什么在策略内强化学习中重要？一项大规模的实证研究，2020。网址 <https://arxiv.org/abs/2006.05990>。

org/abs/2501.12948.  
Etash Guha, Ryan Marten, Sedrick Keh, Negin Raouf, Georgios Smyrnis, Hritik Bansal, Marianna Nezhurina, Jean Mercat, Trung Vu, Zayne Sprague, Ashima Suvarna, Benjamin Feuer, Liangyu Chen, Zaid Khan, Eric Frankel, Sachin Grover, Caroline Choi, Niklas Muennighoff, Shiye Su, Wanjia Zhao, John Yang, Shreyas Pimpalgaonkar, Kartik Sharma, Charlie Cheng-Jie Ji, Yichuan Deng, Sarah Pratt, Vivek Ramanujan, Jon Saad-Falcon, Jeffrey Li, Achal Dave, Alon Albalak, Kushal Arora, Blake Wulfe, Chinmay Hegde, Greg Durrett, Sewoong Oh, Mohit Bansal, Saadia Gabriel, Aditya Grover, Kai-Wei Chang, Vaishal Shankar, Aaron Gokaslan, Mike A. Merrill, Tatsunori Hashimoto, Yejin Choi, Jenia Jitsev, Reinhard Heckel, Maheswaran Sathiamoorthy, Alexandros G. Dimakis, and Ludwig Schmidt. Openthoughts: 推理模型的数据方案, 2025. 网址 <https://arxiv.org/abs/2506.04178>.

# AINLP

- Jian Hu, Xibin Wu, Zilin Zhu, Weixun Wang, Dehao Zhang, Yu Cao, et al. Openrlhf: An easy-to-use, scalable and high-performance rlhf framework. *arXiv preprint arXiv:2405.11143*, 2024.
- Jingcheng Hu, Yinmin Zhang, Qi Han, Daxin Jiang, Xiangyu Zhang, and Heung-Yeung Shum. Open-reasoner-zero: An open source approach to scaling up reinforcement learning on the base model, 2025. URL <https://arxiv.org/abs/2503.24290>.
- Hugging Face. Open rl: A fully open reproduction of deepseek-rl, January 2025. URL <https://github.com/huggingface/open-rl>.
- Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.
- Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*, 2024.
- Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, H erve J egou, and Tomas Mikolov. Fasttext.zip: Compressing text classification models. *arXiv preprint arXiv:1612.03651*, 2016.
- Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets, 2018. URL <https://arxiv.org/abs/1712.09913>.
- Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding rl-zero-like training: A critical perspective. *arXiv preprint arXiv:2503.20783*, 2025.
- Pan Lu, Hritik Bansal, Tony Xia, Jiacheng Liu, Chunyuan Li, Hannaneh Hajishirzi, Hao Cheng, Kai-Wei Chang, Michel Galley, and Jianfeng Gao. Mathvista: Evaluating mathematical reasoning of foundation models in visual contexts. In *International Conference on Learning Representations (ICLR)*, 2024.
- MistralAI. Mistral large 2. <https://mistral.ai/news/mistral-large-2407>, 2024.
- MistralAI. Mistral medium 3. <https://mistral.ai/fr/news/mistral-medium-3>, 2025.
- Michael Noukhovitch, Shengyi Huang, Sophie Xhonneux, Arian Hosseini, Rishabh Agarwal, and Aaron Courville. Asynchronous rlhf: Faster and more efficient off-policy rl for language models. *arXiv preprint arXiv:2410.18252*, 2024.
- Guilherme Penedo, Anton Lozhkov, Hynek Kydl icek, Loubna Ben Allal, Edward Beeching, Agust ın Piqueres Lajar ın, Quentin Gallou edec, Nathan Habib, Lewis Tunstall, and Leandro von Werra. Codeforces cots. <https://huggingface.co/datasets/open-rl/codeforces-cots>, 2025.
- Long Phan, Alice Gatti, Ziwen Han, Nathaniel Li, Josephina Hu, Hugh Zhang, Chen Bo Calvin Zhang, Mohamed Shaaban, John Ling, Sean Shi, et al. Humanity’s last exam. *arXiv preprint arXiv:2501.14249*, 2025.
- David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*, 2024.
- Yousef Saad. *Iterative methods for sparse linear systems*. SIAM, 2003.
- John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.
- Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL <https://arxiv.org/abs/2402.03300>.

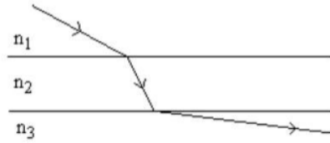
Jian Hu, Xibin Wu, Zilin Zhu, Weixun Wang, Dehao Zhang, Yu Cao 等. Openrlhf: 一个易用、可扩展且高性能的rlhf框架。 *arXiv preprint arXiv:2405.11143*, 2024年。 Jingcheng Hu, Yinmin Zhang, Qi Han, Daxin Jiang, Xiangyu Zhang 和 Heung-Yeung Shum。 Open-reasoner-zero: 一种在基础模型上扩展强化学习的开源方法, 2025年。 网址 <https://arxiv.org/abs/2503.24290>。 Hugging Face。 Open r1: deepseek-r1的完全开源复现, 2025年1月。 网址 <https://github.com/huggingface/open-r1>。 Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aidan Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney 等。 Openai o1系统卡。 *arXiv preprint arXiv:2412.16720*, 2024年。 Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen 和 Ion Stoica。 Livecode bench: 面向代码的大型语言模型的整体且无污染的评估。 *arXiv preprint arXiv:2403.07974*, 2024年。 Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, H erve J egou 和 Tomasz Mikolov。 Fasttext.zip: 压缩文本分类模型。 *arXiv preprint arXiv:1612.03651*, 2016年。 Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer 和 Tom Goldstein。 神经网络损失景观的可视化, 2018年。 网址 <https://arxiv.org/abs/1712.09913>。 Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee 和 Min Lin。 理解r1-zero类训练: 一种关键视角。 *arXiv preprint arXiv:2503.20783*, 2025年。 Pan Lu, Hritik Bansal, Tony Xia, Jiacheng Liu, Chunyuan Li, Hannaneh Hajishirzi, Hao Cheng, Kai-Wei Chang, Michel Galley 和 Jianfeng Gao。 Math vista: 在视觉环境中评估基础模型的数学推理能力。 在 *International Conference on Learning Representations (ICLR)*, 2024年。 MistralAI。 Mistral large 2。 <https://mistral.ai/news/mistral-large-2407>, 2024年。 MistralAI。 Mistral medium 3。 <https://mistral.ai/fr/news/mistral-medium-3>, 2025年。 Michael Noukhovitch, Shengyi Huang, Sophie Khonnoux, Arian Hosseini, Rishabh Agarwal 和 Aaron Courville。 异步rlhf: 更快、更高效的偏离策略强化学习, 用于语言模型。 *arXiv preprint arXiv:2410.18252*, 2024年。 Guilherme Penedo, Anton Lozhkov, Hynek Kydl cek, Loubna Ben Allal, Edward Beeching, Agust n Piqueres Lajar n, Quentin Gallou dec, Nathan Habib, Lewis Tunstall 和 Leandro von Werra。 Codeforces cots。 <https://huggingface.co/datasets/open-r1/codeforces-cots>, 2025年。 Long Phan, Alice Gatti, Ziwen Han, Nathaniel Li, Josephina Hu, Hugh Zhang, Chen Bo Calvin Zhang, Mohamed Shaaban, John Ling, Sean Shi 等。 人类的最后一次考试。 *arXiv preprint arXiv:2501.14249*, 2025年。 David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael 和 Samuel R Bowman。 Gpqa: 一个研究生级别的Google证明问答基准。 在 *First Conference on Language Modeling*, 2024年。 Youcef Saad。 *Iterative methods for sparse linear systems*。 SIAM, 2003年。 John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford 和 Oleg Klimov。 近端策略优化算法。 *arXiv preprint arXiv:1707.06347*, 2017年。 Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu 和 Daya Guo。 Deepseekmath: 推动开放语言模型中数学推理的极限, 2024年。 网址 <https://arxiv.org/abs/2402.03300>。



- Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv:2409.19256*, 2024.
- Shenzhi Wang, Le Yu, Chang Gao, Chujie Zheng, Shixuan Liu, Rui Lu, Kai Dang, Xionghui Chen, Jianxin Yang, Zhenru Zhang, et al. Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for llm reasoning. *arXiv preprint arXiv:2506.01939*, 2025.
- Bo Wu, Sid Wang, Yunhao Tang, Jia Ding, Eryk Helenowski, Liang Tan, Tengyu Xu, Tushar Gowda, Zhengxing Chen, Chen Zhu, et al. Llamarl: A distributed asynchronous reinforcement learning framework for efficient large-scale llm trainin. *arXiv preprint arXiv:2505.24034*, 2025.
- Qiyang Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.
- Xiang Yue, Yuansheng Ni, Kai Zhang, Tianyu Zheng, Ruoqi Liu, Ge Zhang, Samuel Stevens, Dongfu Jiang, Weiming Ren, Yuxuan Sun, Cong Wei, Botao Yu, Ruibin Yuan, Renliang Sun, Ming Yin, Boyuan Zheng, Zhenzhu Yang, Yibo Liu, Wenhao Huang, Huan Sun, Yu Su, and Wenhui Chen. Mmmu: A massive multi-discipline multimodal understanding and reasoning benchmark for expert agi. In *Proceedings of CVPR*, 2024.
- Xiang Yue, Tianyu Zheng, Yuansheng Ni, Yubo Wang, Kai Zhang, Shengbang Tong, Yuxuan Sun, Botao Yu, Ge Zhang, Huan Sun, Yu Su, Wenhui Chen, and Graham Neubig. Mmmu-pro: A more robust multi-discipline multimodal understanding benchmark. In *Proceedings of ACL*, 2025.
- Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models, 2023. URL <https://arxiv.org/abs/2311.07911>.

光明胜、张弛、叶子林峰、吴希彬、张旺、张如、彭阳华、林海滨、吴川。Hybridflow：一个灵活高效的rlhf框架。*arXiv preprint arXiv:2409.19256*, 2024年。王志智、余乐、郜昌、郑楚杰、刘诗轩、卢锐、邓凯、陈雄辉、杨建新、张振如等。超越80/20规则：高熵少数标记推动有效的强化学习以实现llm推理。*arXiv preprint arXiv:2506.01939*, 2025年。吴博、王思德、唐云浩、丁佳、Helenowski Eryk、谭亮、徐腾宇、Gowda Tushar、陈正兴、朱晨等。Llamarl：一个用于高效大规模llm训练的分布式异步强化学习框架。*arXiv preprint arXiv:2505.24034*, 2025年。于启英、张正、朱若飞、袁玉峰、左晓辰、余宇、范甜甜、刘高宏、刘灵君、刘新等。Dapo：一个大规模开源llm强化学习系统。*arXiv preprint arXiv:2503.14476*, 2025年。岳翔、倪元胜、张凯、郑天宇、刘若奇、张戈、Samuel Stevens、姜东福、任伟明、孙玉轩、韦聪、于博涛、袁瑞彬、孙仁良、尹明、郑博远、杨振珠、刘一博、黄文浩、孙欢、苏宇、陈文虎。Mmmu：一个面向专家AGI的庞大多学科多模态理解与推理基准。在*Proceedings of CVPR*, 2024年。岳翔、郑天宇、倪元胜、王宇博、张凯、童胜邦、孙玉轩、于博涛、张戈、孙欢、苏宇、陈文虎、Graham Neubig。在*Proceedings of ACL*, 2025年。Jeffrey Zhou、Lu Tianjian、Swaroop Mishra、Siddhartha Brahma、Basu Sujoy、Luan Yi、Zhou Denny、侯乐。大语言模型的指令遵循评估, 2023年。网址<https://arxiv.org/abs/2311.07911>。

**Problem:**



A ray of light passes from medium 1 to medium 2 to medium 3 as shown in the diagram. What may be concluded about the speed of light in each medium?

- A.  $v_1 > v_2 > v_3$
- B.  $v_1 > v_3 > v_2$
- C.  $v_2 > v_1 > v_3$
- D.  $v_2 > v_3 > v_1$
- E.  $v_3 > v_1 > v_2$

**Model Generation:**

<think> First, I need to understand the behavior of light as it passes through different mediums. According to Snell's Law, the speed of light changes when it enters a medium with a different refractive index. The refractive index ( $n$ ) of a medium is inversely related to the speed of light ( $v$ ) in that medium, meaning that as the refractive index increases, the speed of light decreases.

Looking at the diagram: The light beam starts from medium 1 to medium 2, bending towards the normal line. This indicates that medium 2 has a higher refractive index than medium 1 because light bends towards the normal when it slows down (entering a denser medium). The light beam that passes from medium 2 to medium 3, bending away from the normal line. This indicates that medium 3 has a lower refractive index than medium 2 because light bends away from the normal when it speeds up (entering a less dense medium).

Now, let's analyze the speed of light in each medium: In medium 1, the speed of light is  $v_1$ . In medium 2, since the light bends towards the normal when entering from medium 1, medium 2 has a higher refractive index than medium 1. Therefore, the speed of light in medium 2 is  $v_2$ , is less than in medium 1, so  $v_1 > v_2$ . In medium 3, since the light bends away from the normal when entering from medium 2, medium 3 has a lower refractive index than medium 2. Therefore, the speed of light in medium 3,  $v_3$ , is greater than in medium 2, so  $v_3 > v_2$ .

Now, we need to compare the speeds  $v_1$ ,  $v_2$ , and  $v_3$ . From the above analysis,  $v_1 > v_2$  and  $v_3 > v_2$ . But we also need to determine the relationship between  $v_1$  and  $v_3$ . Since medium 3 has a lower refractive index than medium 1 (because light bends away from the normal when transitioning from medium 2 to medium 3, and we already know medium 2 has a higher refractive index than medium 1), it implies that medium 3 is less dense than medium 1. Therefore, the speed of light in medium 3 will be greater than in medium 1.

So, in sum:  $v_3 > v_1 > v_2$

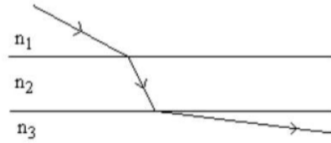
Looking at the options provided: A.  $v_1 > v_2 > v_3$  B.  $v_1 > v_3 > v_2$  C.  $v_2 > v_1 > v_3$  D.  $v_2 > v_3 > v_1$  E.  $v_3 > v_1 > v_2$  The correct option that matches our conclusion is A.

</think>

**Final Answer: A**

**Figure 14:** A physics multimodal problem and its solution generated by Magistral Medium.

Problem:问题:



一道光线从介质1经过介质2到介质3，如图所示。可以得出关于每个介质中光速的什么结论？

- A.  $v_1 > v_2 > v_3$
- B.  $v_1 > v_3 > v_2$
- C.  $v_2 > v_1 > v_3$
- D.  $v_2 > v_3 > v_1$
- E.  $v_3 > v_1 > v_2$

Model Generation:模型生成:

<首先，我需要了解光在通过不同介质时的行为。根据斯涅尔定律，当光进入折射率不同的介质时，其速度会发生变化。介质的折射率（ $n$ ）与该介质中光的速度（ $v$ ）成反比，也就是说，折射率越高，光速越低。

观察图示：光束从介质1射向介质2，向法线方向弯折。这表明介质2的折射率高于介质1，因为当光线减速（进入更密的介质）时会向法线弯折。通过介质2到介质3的光束则远离法线弯折。这表明介质3的折射率低于介质2，因为当光线加速（进入较稀的介质）时会远离法线。

现在，让我们分析每个介质中的光速：在介质1中，光速为 $v_1$ 。在介质2中，由于光在从介质1进入时向法线弯折，介质2的折射率高于介质1。因此，介质2中的光速为 $v_2$ ，小于介质1中的光速，所以 $v_1 > v_2$ 。在介质3中，由于光在从介质2进入时远离法线弯折，介质3的折射率低于介质2。因此，介质3中的光速 $v_3$ ，大于介质2中的光速，所以 $v_3 > v_2$ 。

现在，我们需要比较速度 $v_1$ 、 $v_2$ 和 $v_3$ 。根据上述分析， $v_1 > v_2$   $v_3 > v_2$ 但我们还需要确定 $v_1$ 和 $v_3$ 之间的关系。由于介质3的折射率低于介质1（因为当光从介质2转变到介质3时，光偏离法线，而我们已经知道介质2的折射率高于介质1），这意味着介质3的密度比介质1小。因此，光在介质3中的速度将大于在介质1中的速度。

所以，总结一下： $v_3 > v_1 > v_2$

查看提供的选项：A.  $v_1 > v_2 > v_3$  B.  $v_1 > v_3 > v_2$  C.  $v_2 > v_1 > v_3$  D.  $v_2 > v_3 > v_1$  E.  $v_3 > v_1 > v_2$  与我们的结论相符的正确选项是 A。

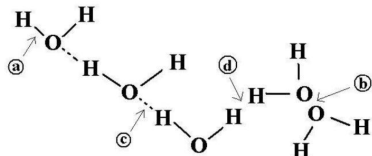
</think>

最终答案：A

Figure 14: A physics multimodal problem and its solution generated by Magistral Medium.

**Problem:**

Which arrow points to a hydrogen bond?

**Model Generation:**

<think> Alright, I have this image showing a molecular structure with arrows labeled a, b, c, and d. The question is asking which arrow points to a hydrogen bond. Let's break this down.

First, I need to understand what a hydrogen bond is. A hydrogen bond is a type of attractive interaction between a hydrogen atom and an electronegative atom like oxygen, nitrogen, or fluorine. In this case, since we have water molecules (H<sub>2</sub>O), the hydrogen bonds would be between the hydrogen atoms and the oxygen atoms of different water molecules.

Looking at the image:

- Arrow a points to a bond between a hydrogen atom and an oxygen atom within the same water molecule. That's a covalent bond, not a hydrogen bond.
- Arrow b points to another bond within a single water molecule, which is again a covalent bond.
- Arrow c points to a bond between a hydrogen atom of one water molecule and an oxygen atom of another water molecule. This looks like a hydrogen bond because it's between different water molecules.
- Arrow d points to a bond between hydrogen and oxygen within a single water molecule, so that's another covalent bond.

From this, it seems that arrow c is pointing to the hydrogen bond between the hydrogen of one water molecule and the oxygen of another water molecule.

But just to be sure, I should double check. Hydrogen bonds typically occur between molecules, not within the same molecule, and they involve a hydrogen atom being attracted to a highly electronegative atom like oxygen. Arrow c is indeed showing this intermolecular interaction between an H of one molecule and O of another molecule.

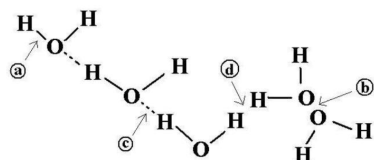
</think>

The image shows four arrows labeled a, b, c, and d. Arrow c points to a bond between a hydrogen atom in one water molecule and an oxygen atom in another water molecule. This is characteristic of a hydrogen bond, which occurs between different molecules, not within the same molecule.

Final Answer: c

**Figure 15:** A chemistry multimodal problem and its solution generated by Magistral Medium.

Problem:问题:  
哪个箭头指向氢键?



Model Generation:模型生成:

<think> 好的,我有一张显示分子结构的图片,箭头标记为 a、b、c 和 d。问题是问哪个箭头指向氢键。让我们拆解一下。

首先,我需要了解什么是氢键。氢键是一种氢原子与氧、氮或氟等电负性原子之间的吸引相互作用。在这种情况下,由于我们有水分子(H<sub>2</sub>O),氢键将发生在不同水分子的氢原子和氧原子之间。

看着图片:

- 箭头 a 指向同一水分子内氢原子和氧原子之间的键。这是一个共价键,而不是氢键。
- 箭头 b 指向单个水分子内的另一条键,这也是一条共价键。
- 箭头 c 指向一个氢原子与另一个水分子的氧原子之间的键。这看起来像一个氢键,因为它发生在不同的水分子之间。
- 箭头 d 指向单个水分子中氢和氧之间的键,因此那是另一种共价键。

由此看来,箭头 c 指向的是一个水分子的氢原子与另一个水分子的氧原子之间的氢键。

但为了确保无误,我应该再确认一下。氢键通常发生在分子之间,而不是在同一分子内部,它涉及一个氢原子被吸引到一个高电负性的原子,比如氧。箭头c确实显示了一个分子的氢原子与另一个分子的氧原子之间的分子间相互作用。

</think>

图片显示四个箭头,标记为 a、b、c 和 d。箭头 c 指向一个氢原子与另一个水分子中的氧原子之间的键。这是氢键的特征,氢键发生在不同的分子之间,而不是在同一分子内部。

最终答案: c

图15: 由Magistral Medium生成的化学多模态问题及其解决方案。

**Problem:**

What is leading to the crinkling of this leaf's veins?

- A. Bacterial pathogen
- B. I don't know and I don't want to guess
- C. Fungal pathogen
- D. Oomycete (watermold) pathogen
- E. Physiological condition, no pathogen involved.

**Model Generation:**

<think> The image shows a close-up of a leaf with some crinkling and discoloration along the veins. To determine the cause of the crinkling, we need to consider the different possibilities given in the options.

Firstly, let's rule out some options: Bacterial pathogens (Option A) usually cause symptoms such as water-soaked lesions, wilting, and bacterial ooze, not typically the crinkling seen here. Fungal pathogens (Option C) often cause spots, mildews, or powdery substances on leaves, which doesn't match the appearance here. Oomycete (watermold) pathogens (Option D) often lead to water-soaked lesions and a more generalized blight rather than vein-specific crinkling.

Now, let's consider the remaining options: Option B ("I don't know and I don't want to guess") is not a valid answer as we are tasked to determine the cause of the crinkling. Option E (physiological condition) could be a possibility as it involves conditions like nutrient deficiencies or environmental stress that can lead to crinkling without pathogen involvement.

The appearance of the leaf shows crinkling and some discoloration along the veins, which can often be attributed to physiological conditions. These might include nutrient deficiencies, water stress, or other environmental factors that do not involve pathogens.

Given the visual clues and the typical symptoms associated with different causes, the most likely cause of the crinkling in this leaf's veins is a physiological condition rather than a pathogen.

</think>

Summary: The image shows a leaf with crinkled veins and some discoloration. After considering the typical symptoms associated with various causes, it's most likely that the crinkling is due to a physiological condition rather than a bacterial, fungal, or oomycete pathogen. Therefore, the correct answer is:

Final Answer: E

**Figure 16:** A biology multimodal problem and its solution generated by Magistral Medium.

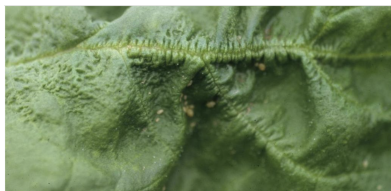


Problem:问题:

导致这片叶子脉络皱缩的原因是什么?

A. 细菌性病原体 B. 我不知道, 也不想猜 C.

真菌性病原体 D. 霜霉菌 (水霉) 病原体 E. 生理状态, 无病原体参与。



Model Generation:模型生成:

<思考> 这张图片显示了一片叶子的特写, 叶脉上有一些皱折和变色。为了确定皱折的原因, 我们需要考虑选项中给出的不同可能性。

首先, 让我们排除一些选项: 细菌病原体 (选项A) 通常引起水浸状病斑、萎蔫和细菌渗出, 而不太可能出现这里看到的皱缩。真菌病原体 (选项C) 常在叶片上引起斑点、霉菌或粉状物, 这与这里的外观不符。卵菌 (水霉) 病原体 (选项D) 通常导致水浸状病斑和更广泛的枯萎, 而不是特定于脉络的皱缩。

现在, 让我们考虑剩下的选项: 选项B (“我不知道, 也不想猜”) 不是一个有效的答案, 因为我们的任务是确定皱折的原因。选项E (生理状况) 可能是一种可能性, 因为它涉及营养缺乏或环境压力等条件, 这些都可能導致皱折而不涉及病原体。

叶子的外观显示出皱缩和沿着叶脉的一些变色, 这通常可以归因于生理条件。这些可能包括营养缺乏、水分压力或其他不涉及病原体的环境因素。

根据视觉线索和与不同原因相关的典型症状, 这片叶脉皱折的最可能原因是生理状况, 而非病原体。

</think>

摘要: 图片显示一片叶子, 叶脉皱缩并伴有一些变色。在考虑了与各种原因相关的典型症状后, 最有可能的原因是皱缩是由生理状况引起的, 而非细菌、真菌或卵菌病原体。因此, 正确答案是:

最终答案: E

**Figure 16:** A biology multimodal problem and its solution generated by Magistral Medium.