

---

# 庄严

---



## Abstract

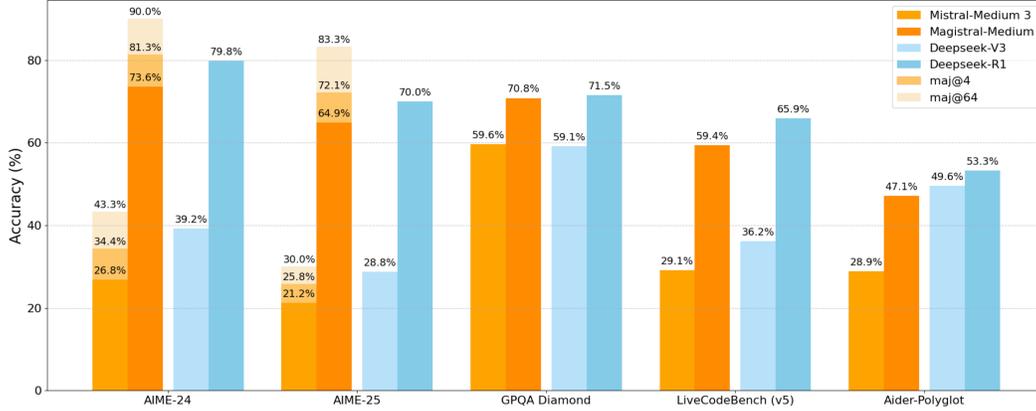
我们引入了 Magistral，这是 Mistral 的第一个推理模型和我们自主开发的可扩展强化学习 (RL) 流程。与依赖现有实现和从先前模型提取的 RL 轨迹不同，我们采用自下而上的方法，仅依赖于我们自己的模型和基础设施。值得注意的是，我们展示了一套能够让我们探索纯粹 RL 训练 LLM 极限的架构，提出了一种简单的方法来强制模型的推理语言，并展示仅基于文本数据进行 RL 可以维持初始检查点的大部分能力。我们发现文本上的 RL 可以保持或改善多模态理解、指令遵循和函数调用。我们展示了 Magistral Medium，专用于推理且仅通过 RL 训练于 Mistral Medium 3 之上，并开源了 Magistral Small (Apache 2.0)，其中进一步包含了来自 Magistral Medium 的冷启动数据。

## 1 引言

增强大语言模型 (LLMs) 的推理能力已成为现代人工智能研究的关键前沿。推理模型如 o1 [Jaech et al., 2024] 与经典聊天机器人有很大不同，它利用更长的思维链来提高复杂任务的性能。DeepSeek-AI et al. [2025] 的开创性工作向社区提供了关于可验证奖励的强化学习 (RLVR) 配方的重要见解，以便大规模创建推理模型。

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，并详细概述我们提出的 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训练 Magistral Medium，而没有从现有的推理模型中蒸馏，从而在 AIME-24 (pass@1) 上实现了近 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打断它们，实现了快速和连续的强化学习训练，从而在效率与策略性之间取得了平衡。
- 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，其中包括用用户的语言撰写思维链和最终响应。
- 我们提供了对现有 RLVR 文献的洞察，包括是否强化学习 (RL) 可以在小模型的蒸馏 SFT 基线上有所改进，或与其相悖。我们还展示了基于多模态模型的多模态推理能力在使用文本数据的在线 RL 中出现的情况。我们还分享了我们不成功实验的结果。
- 我们在 Apache 2 许可证<sup>1</sup>下发布了 Magistral Small (24B) 的权重。



**Figure 1:** 法官中等模型在常见推理基准上的表现。我们强调所提出的 RLVR 框架的优势，该框架在没有任何冷启动推理线索的情况下，使 AIME-24 (pass@1) 的成绩比最初的 Mistral Medium 3 检查点提高了 50%。我们与 [DeepSeek-AI et al., 2025] 的类似结果进行比较，后者显示了从 DeepSeek-v3 到 DeepSeek-R1 (1 月 25 日) 的 RL 改进。法官中等模型通过多数表决在 AIME-24 上达到了 90% 的准确率。

本文内容组织如下：第 2 节详述了我们使用的 RL 算法，以及用于指导推理模型的语言和格式设计选择；第 3 节介绍了我们可扩展的基础设施，该基础设施支持在大型 GPU 集群上的高效训练；第 4 节讨论了用于高效和有效训练的数据选择过程；第 5 节展示了 Magistral 在推理和多语言基准上的性能；第 6 节展示了为了激励训练选择而进行的消融实验；第 7 节介绍了基于 PCA 的关于模型权重在 RL 过程中的轨迹的研究，并证明了基于文本数据进行 RL 保留甚至提升了多模态能力，同时包括那些对 Magistral 效果不佳的方法；第 8 节展示了可以通过蒸馏然后进行 RL 来训练一个能够与 R1 媲美的模型，虽然我们没有使用这种方法来进行 Magistral Medium 的训练；最后，在第 9 节中，我们总结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，以确保训练的稳定性（第 2.1 节）以及我们的训练奖励，以提高数学和编码能力，同时确保模型符合适当的格式、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 et al., 2024] 作为我们的强化学习算法。与 PPO [Schulman et al., 2017] 不同，GRPO 消除了对“评判模型”的需求，并且通过使用策略中每个提示的多次生成的平均奖励来计算优势计算的基准。具体来说，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提取的查询， $o$  代表模型的生成， $\varepsilon$  是 PPO 剪切阈值， $\beta$  是 KL 惩罚系数， $D_{\text{KL}}$  表示当前策略  $\pi_\theta$  与参考策略  $\pi_{\text{ref}}$  之间的 Kullback-Leibler 散度。相对优势或组归一化优势由  $\hat{A}_{i,t} = \frac{r_{i,t} - \mu}{\sigma}$  给出，其中  $\mu$  和  $\sigma$  是在一个组内计算的奖励的平均值和标准偏差。基于先前为推理任务适配 GRPO 的工作 [Yu et al., 2025, Liu et al., 2025, Hu et al., 2025]，我们引入了若干改进：

消除 KL 散度。KL 散度惩罚限制了在线策略过多偏离参考策略，有助于保持与初始模型的一致性。然而，在 GRPO 中，无论如何策略都会显著偏离，而为了进行 KL 计算而维护参考模型的副本造成了计算成本，我们认为这是不合理的。我们完全去掉了 KL 惩罚。

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

损失归一化。为了避免在一个组内生成的长度偏差，我们通过首先对所有标记和所有生成实例的标记损失进行求和，然后再除以组  $\sum_{i=1}^G |o_i|$  中生成实例的总长度来进行损失归一化。

优势归一化。我们估计每个标记的优势为  $\hat{A}_{i,t} = \hat{A}_i = r_i - \mu$ ，其中  $\mu$  是组内奖励的平均值。根据 Andrychowicz et al. [2020]，我们在每个小批次中附加地将优势标准化为  $\hat{A}_{i,t}^{\text{norm}} = (\hat{A}_i - \hat{A}^{\text{mean}}) / \hat{A}^{\text{std}}$ ，其中  $\hat{A}^{\text{mean}}$  和  $\hat{A}^{\text{std}}$  分别是小批次中序列化计算的优势  $\hat{A}_i$  的均值和标准差。

放宽信任域的上限。我们允许模型探索罕见但潜在有见解的推理步骤，防止确定性策略的生成。我们采用 Clip-Higher [Yu et al., 2025] 策略来解决熵崩溃问题。在标准 GRPO 中， $\epsilon$  剪辑通过限制低概率的 token 概率增加来限制探索，阻碍了罕见但重要的推理路径的增强。通过将上限剪辑阈值增加到  $\epsilon_{\text{high}}$ ，低概率的 token 有更多的增长空间，从而提高了输出的熵和多样性，改善了推理探索。我们发现，对  $\epsilon_{\text{high}}$  的精心调整对于在 RL 运行中保持稳定性至关重要。我们在训练过程中将其调整在 0.26 到 0.28 之间，以保持组熵的稳定性。

消除非多样化组。所有生成结果完全正确或错误的组没有优势，因此对批次损失没有贡献。这导致梯度变小，并增加了对噪声的敏感性。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，包括所有以红色标出的修改项，是

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

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

## 2.2 奖励塑造

选择适当的奖励对于确保 RL 算法有效工作至关重要。在训练过程中，模型生成结果根据四个方面进行评估：格式、正确性、长度和语言一致性，我们将在下文中描述这些方面。

### 2.2.1 格式化

对于数学和代码问题，我们指示模型遵循特定格式，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，并必须包括相应的 `</think>` 标签。(ii) 响应中应正好存在一组这样的标签。
2. 数学响应：对于数学输出，响应必须包括答案部分中的最终答案，并以 `\boxed { }` 括起来，跟随在 `</think>` 标签之后。
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和编程语言说明格式化的 markdown 块。

如果不满足这些条件中的任何一个，将导致奖励为 0，并且不会进一步评分。否则，该反应将获得 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，我们将提取模型解决方案，并使用验证器评估其正确性。

数学正确性。最终答案从解决方案中最后一个 `\boxed { }` 提取出来，并通过基于规则的验证器与参考答案进行比较。它会对标准答案和生成的答案进行标准化，以正确奖励具有不同语法的语义相同的响应。我们利用不同解析器和 SymPy<sup>2</sup> 的组合来评估输出并将其与原始标准答案进行比较。如果答案正确，则给予额外奖励 0.9，使总奖励达到 1.0。

代码正确性。代码从答案部分的第一个 markdown 代码块中提取。如果代码是用 C++ 编写的，则使用 C++20 标准进行编译，设定的超时时间为 10 秒。我们预先编译了在竞赛编程中常用的 `bits/stdc++.h` 标准库头文件，以加快编译过程。我们从可用的测试用例中随机选

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

### 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 的系统提示。系统提示详细说明了模型的格式和语言指南。相同的系统提示用于数学和编码问题。

择 20 个测试，确保在给定的响应组中使用相同的测试。然后针对这些测试执行代码，每个测试有 4 秒的超时时间和 300 MB 的内存限制。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

根据 [Yu et al., 2025]，我们使用软长度惩罚来提醒模型，最大完成长度的硬限制接近。我们固定两个长度  $l_{\max}$  和  $l_{\text{cache}}$ ，并计算长度处罚为

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

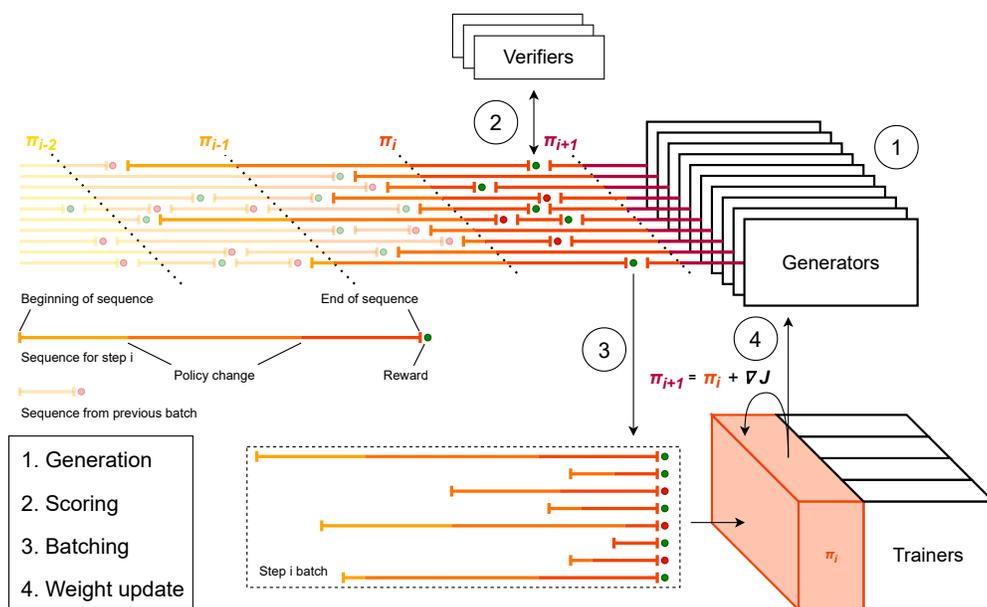
### 2.2.4 语言一致性奖励

Magistral 的核心设计原则是让它使用与用户相同的语言进行推理。在未经过任何处理的数学和编码问题上的强化学习通常会导致混合语言的模型响应。在没有语言限制的初步实验中，我们经常观察到输出混合了英语、中文和俄语词汇。虽然这些输出是连贯的，但从用户的角度来看，这是不理想的。

为了防止语言切换，我们将用英文撰写的 10 个 % 问题翻译成以下语言：法语、西班牙语、意大利语、德语、中文和俄语。在计算对话的奖励时——一个由（问题、思考和答案）组成的三元组——我们首先通过去除 LaTeX 内容和代码块来规范这些组成部分，然后应用 fastText 分类器 [Joulin et al., 2016] 进行分类。如果分类器表明三个部分都使用相同的语言，我们会给予额外的 0.1 奖励。

这些简单的修改足以使模型能够紧密地跟随用户的语言，且极少出现代码转换，同时保持在推理任务上的性能。尽管我们仅将原始的英语问题翻译成几种语言，但我们观察到模型可以成功地在任意语言中生成思维链。

系统提示。我们在系统提示中指定了格式和语言要求，可以在图 2 中找到。我们发现，RL 训练对我们使用的系统提示非常敏感。例如，系统提示的 `Be as casual and as long as you want` 部分增加了模型的熵，从而改善了模型的探索。



**Figure 3:** 在线训练流程。1) 生成器持续地从输入数据源输出对提示的补全。2) 每当一个补全完成后, 它会被发送到适当的验证器。3) 每个序列被根据预设的排列发送到不同的数据并行组, 直到每个数据并行组有足够的序列来形成一个批次。4) 进行一次梯度更新操作, 训练器和生成器随后被更新。在生成器中, 权重在生成中期被替换, 这意味着正在进行的生成将继续使用略微过时的键值缓存, 因为我们不刷新缓存。由于模型在训练器和生成器上都驻留在 GPU 上, 权重通过 NCCL 传输以实现最佳性能。模型权重被动态整合以适应训练器和生成器之间不同的切片拓扑。

### 3 基础设施

在本节中, 我们介绍我们的在线训练基础设施。我们采用了一种与之前多项工作中提出的类似的分布式 RL 训练系统, 该系统协调三种类型的工作者:

- 训练者维护模型权重的主要副本并进行梯度更新。
- 生成器执行“展开”, 使用最新策略返回来自训练提示的完成和对数概率。
- 验证者评估生成器生成的完成情况并返回奖励 (详细信息见第 2.2 节)。

**分布式强化学习的挑战。** 生成器是总计算量中的重要部分, 也是在线 RL 过程中独特的一部分。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极不均匀, 并随着训练过程的变化而变化: 最长的完成可能比最短的要花费长达 5 倍的时间。系统的主要限制之一是不对序列长度引入任何偏差: 即使较短的完成更快完成, 完成长度的分布也必须与训练数据完全一致。一个相互竞争的目标是尽快更新生成器的权重。我们希望生成尽可能遵循策略, 但也希望生成器可以在不等待彼此或训练者的情况下运行。

**异步生成。** 为了在没有任何近似的情况下进行训练, 我们可以顺序处理批次: 在一个批次上启动生成器, 等待所有序列完成, 更新训练员和生成器的模型权重, 然后重复。然而, 由于异构完成时间, 该方法会导致生成器空闲和流水线效率低下。相反, 我们优先考虑效率, 并在最大吞吐量下不间断地操作生成器, 从不等待训练员的完成。我们不断从生成器中收集, 验证它们, 并更新训练员。在这些更新之后, 训练员通过 NCCL 将新权重发送到生成器, 而不会舍弃当前正在生成的飞行中的序列。从 GPU 广播权重到 GPU 十分重要, 因为即使在大型模型和大型世界规模的情况下, 它也将单次更新所需的时间减少到不到 5 秒。我们在图 3 中展示了这个过程。

当为一个单一的提示生成解决方案时, 它可能会多次更新模型权重, 以反映训练者的最新改进。在它被完全处理并发送给训练者时, 模型权重可能已经更新了好几次, 但最新的标记总是按照策略生成的。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稍微

过时，因为它们是由模型的以前版本计算得出的。为了性能，我们发现重新计算键值缓存不是必要的，这可能是由于损失函数 [Schulman et al., 2017] 固有的离策略修正。

**训练器优化。** 我们将一个批次定义为固定数量的生成完成，而不是固定数量的标记。生成器根据预设的排列，将每个已完成的生成发送到随机的训练器排名。当每个数据并行排名收到足够的生成完成以形成一个批次时，执行梯度更新。如果训练器成为瓶颈，例如在生成长度仍然较短的早期训练中，我们会将接收到的生成积累到一个有固定大小限制的阻塞队列中，以控制非策略行为的程度。一个批次可能会被分成小批次，以执行多次优化步骤（见第 6.3 节）。每个小批次有固定数量的生成完成，但标记数量可变，因此进一步分为固定标记大小的微批次。由于我们是在微批次上积累梯度，样本的顺序并不重要。我们利用这一特性实现了一种贪婪聚合算法，将序列按大小降序排序并尝试一个个放入空闲的微批次，如果没有则新建。这确保了每个小批次训练工作者的工作负载均匀，减少了 19% 的填充。

## 4 数据整理

我们将自己限制在可验证解的问题上；我们使用其解是数值答案或表达式的数学问题，并编写带有相关测试的代码问题。我们在此描述我们应用的广泛过滤。

### 4.1 数学

**格式过滤。** 我们从一个大型但噪声较大的问题集合开始，该集合约有 70 万样本。我们首先对数据进行了全面的预处理和过滤，以确保所有问题都是完整的，并且最终答案是准确且可以使用基于规则的系统进行验证的。特别是，我们过滤了那些以证明为基础和具有多部分的难以验证正确性的问题。此外，我们将多项选择题重新表述为基于陈述的问题，以实现更强的验证能力和增加的难度。

我们实施了一个两阶段的过滤流程，以整理一个“恰到好处”难度级别的问题数据集，即对模型来说既不太容易也不太困难。首先，我们使用 Mistral Large 2 [MistralAI, 2024] 进行了初步的难度评估，通过对每个问题采样 16 个解并移除那些从未解决或解决率极高的问题。然后使用这个初步整理的问题集合，通过我们的在线 RL 流程训练了一个 24B 模型，产生了一个小但有能力的检查点，我们仅用于评分。

在第二阶段，使用这个经过 RL 训练的更强模型重新评估整个原始数据集。我们再次为每个问题抽取 16 个回应，筛选出最简单和仍未解决的问题。然后进一步过滤掉可能不正确的题目，即多数样本具有相同最终答案但与“参考答案”不一致的问题。这是因为当模型持续得出与参考解相悖的一致意见时，问题本身更可能有错误的参考答案。

这种两阶段的方法至关重要，因为单靠初始的较弱的 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分类为无法解决，从而丢弃这些问题。通过使用一个更强、经过 RL 训练的模型进行第二次处理，我们确保这些有价值且具有挑战性的训练样例被准确地评估并保留下来。

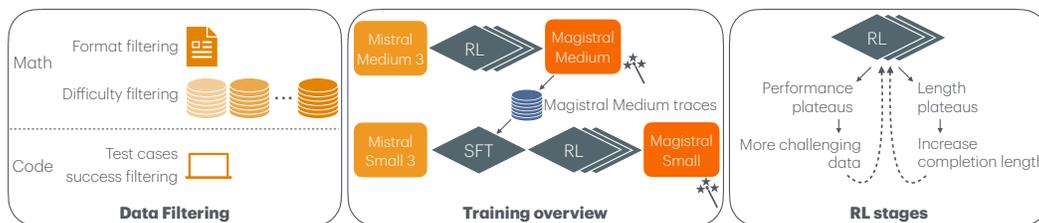
Table 1: 不同筛选阶段后数学训练样本的数量。

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描述，并在可用时包括正确的解决方案和相关的测试。对于训练过程，我们需要问题描述和每个问题的大量正确测试。为此，我们首先移除没有解决方案和没有足够测试的问题。然后，我们在所有可用测试上执行每个解决方案，并丢弃测试结果不一致的测试。对于那些有足够一致性但没有一个解决方案成功通过的测试，我们假设测试不正确，并更新测试以反映解决方案输出中最常见的结果。在没有测试的代码问题情况下，我们生成额外的测试并对其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，这两种语言是竞赛编程中常用的。这一过程产生了一个包含 3.5 万个代码问题的数据集。



**Figure 4:** 这篇文章中讨论的过滤、训练和强化学习阶段的概述。我们在 Mistral Medium 3 上进行强化学习得到 Magistral Medium。我们使用该模型为一组多样的提示生成答案。我们使用这些生成的轨迹微调 Mistral Small 3，然后执行强化学习以获得 Magistral Small。

## 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) 在一个大型基础模型上仅使用纯粹的强化学习可以达到什么程度？(ii) 给定一个强大的教师模型，如何获得最强的轻量级模型？为此，我们在 Mistral Medium 3 [MistralAI, 2025] 之上使用纯强化学习训练了 Magistral Medium；而 Magistral Small 则开始于从 Magistral Medium 导出的 SFT 痕迹。

### 5.1 评估基准和基线

我们报告了在数学、编程和 STEM 领域评估能力的基准测试结果。对于数学，我们提供了美国邀请数学考试基准 (AIME'24, AIME'25) 和 MATH 数据集上 [Hendrycks et al., 2021] 的结果。在编程中，我们包括了 LiveCodeBench (版本 v5 和 v6) [Jain et al., 2024]，以及 Aider Polyglot [Gauthier, 2024]。对于 STEM，我们报告了基于 GPQA 数据集 [Rein et al., 2024] 的结果。此外，我们还报告了我们在《人类最后的考试》 [Phan et al., 2025] 中的文本问题上的结果，该考试包括了涉及数学、人文和自然科学的数十个学科的 2,500 道题目。对于所有评估任务，我们将温度设定为 0.7，对于数学评估和 GPQA 使用 top-p 为 1.0，对于编程任务则为 0.95。AIME 和 LiveCodeBench 的最大标记长度设置为 40k，其他评估的设置 32k。

作为基线，我们包括来自 [DeepSeek-AI et al., 2025] 的结果，该结果报告了大规模 RL 训练的可比数据点，包括有和没有从推理模型的轨迹上进行 SFT 的情况。

### 5.2 杰出媒介——从头开始推理 RL

在此，我们的目标是通过训练一个没有任何“冷启动”（即通过推理过程的蒸馏进行推理启发）的模型来评估我们 RL 栈的质量。我们使用了 Mistral Medium 3 Instruct [MistralAI, 2025] 作为此次运行的起始检查点。训练是在多个阶段中完成的，每个阶段采用不同的超参数。特别是，这些阶段的设计确保始终满足以下标准：

1. 数据集并不简单。随着模型性能的提高，我们增加了数据的难度。通过加入更复杂的数据（这些数据在早期阶段被过滤掉）或从数据中移除已经完全解决的问题，构建更难的数据划分。
2. 生成长度不会停止增长。为了防止生成长度停滞，我们增加了最大允许完成长度和不受长度惩罚的最大完成长度  $l_{\max} - l_{\text{cache}}$ （参见第 2.2.3 节）。我们将  $l_{\max} - l_{\text{cache}}$  增加为 16k  $\rightarrow$  24k 和 24k  $\rightarrow$  32k 的两倍。
3. KV 缓存的内存负担不太大。随着生成长度的增加，与 KV 缓存相关的内存使用量也增加。为了解决这个问题，我们缩小了并发请求总数  $n_{\text{async}}$ 、批大小  $n_{\text{batch}}$  和小批量大小  $n_{\text{minibatch}}$ 。批大小的影响在第 6.3 节中讨论。在训练期间，我们曾两次减少批大小，分别是 8k  $\rightarrow$  4k 和 4k  $\rightarrow$  2k。

表格 2 显示了纯 RL 训练的 Magistral Medium 的结果，并与 [DeepSeek-AI et al., 2025] 的类似实验进行了比较。我们发现，仅凭我们的 RL 流程就能够使 AIME '24 的准确率 (pass@1) 提高近 50%，在 LiveCodeBench (v5) 上提高 30%。

### 5.3 Magistral Small – 推理上的强化学习自监督微调引导

鉴于我们在 Magistral Medium 中有一个强大的“教师”模型，接下来我们将探讨如何训练出可能的最强的学生模型。为此，我们训练了 Magistral Small，它是通过 Magistral Medium 中的 SFT 痕迹进行“冷启动”的。

与纯粹的 RL 训练（依赖于少量极其干净和困难的训练点，见第 4 节）相比，我们发现提示的多样性对推理的冷启动非常重要。我们首先从 Magistral Medium 的 RL 训练中提取出具有正确答案的轨迹，排除那些来自早期步骤且具有简短 CoT 的轨迹。我们还通过限制每个问题的生成次数来保持问题的难度水平混合，以避免收集的轨迹偏向更容易的问题，并且对通过率较低的问题进行上采样。

我们通过在一大组多样化的提示上生成我们 Magistral Medium 的回应来增强该 SFT 冷启动数据，这些提示来源于 OpenThoughts [Guha et al., 2025] 和 OpenR1 [Hugging Face, 2025, Penedo et al., 2025] 的代码子集。我们在顶部进行了额外筛选，并保留了一部分提示。这为我们提供了一个具有混合难度的推理数据集。我们还包括了 10% 个数据点用于一般指令调优，以保留非推理能力。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，并选择了在 AIME’24 上表现最佳的检查点作为后续 RL 阶段的初始检查点。

然后，我们使用 RL 训练这个 SFT 检查点，批量大小为 2048 个序列，最大非惩罚完成长度  $l_{\max} - l_{\text{cache}}$  为 32k。我们为生成使用了 1.0 的采样温度，因为它在避免低温度所见的多样性缺乏与高温生成出的不连贯输出之间提供了最佳平衡。我们使用了  $\epsilon_{\text{high}}$  为 0.3，以鼓励探索，因为冷启动模型产生了熵远低的响应。

表格 3 显示了 24B 模型在三种不同范式下训练的性能：仅使用 SFT；仅使用 RL；以及在冷启动检查点基础上使用 RL。这里，与 [DeepSeek-AI et al., 2025] 的发现相反，我们发现即使在一个较小的基础模型上，使用 RL 也能在从较大教师模型中蒸馏之外获得显著提升。这强调了本工作中引入的 RL 堆栈的优势。

**Table 2:** 仅使用 RL 训练的 Magistral Medium 的结果。为了减少方差，我们在 AIME 上取 64 次运行的平均值（显示为 pass@1/maj@64），在 LiveCodeBench 上取 16 次运行的平均值。人类的最后一项仅评估文本子集。

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 多语言基准测试

为了评估 Magistral 的多语言能力，我们使用多种语言与 Magistral Medium 进行交互，以检查它是否能够在用户的语言中进行推理和回答。我们还测试了 Magistral Medium 在 AIME 2024 基准测试的多语言版本（包括法语、西班牙语、德语、意大利语、俄语和中文）上的表现。这些多语言版本是通过将问题从英语翻译成每种语言创建的。结果如表 4 所示。我们发现，与英语相比，该模型在多语言版本上表现差 4.3-9.9%，可能相当于实际 AIME 测试中的 1-3 个问题，可能因为我们限制了推理的语言。这种退化与基础模型的情况大致相似。需要注意的是，在多语言基准测试中，所有推理和最终回答都是在输入语言中进行的（即，不是英语）。

**Table 3:** Magistral Small 在各种基准测试中与不同的训练设置相比的表现。我们报告了三个不同的 24B 模型的性能：细调在 Magistral Medium 推理轨迹上的 Mistral Small 24B (SFT)，从头开始用 RL 训练的 Mistral Small 24B (仅 RL)，以及细调在 Magistral Medium 轨迹上并随后用 RL 增强的 Mistral Small 24B (SFT + RL)，这也是最终的 Magistral Small。我们观察到，将推理轨迹的细调与 RL 结合起来，能导致最佳性能。在评估 Humanity’s Last Exam 时，仅考虑了文本子集。

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

**Table 4:** Magistral Medium 在 AIME 2024 基准的多语言版本上的 pass@1 性能。

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，以研究当仅对一种模态进行 RL 时会发生什么，将 RL 与蒸馏 SFT 基线进行比较，并阐明我们必须考虑的两个训练选择，即批量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）上进行训练并在另一个领域进行评估来调查我们模型的跨领域泛化能力。具体来说，我们在 24B 模型上进行了两个实验：一个是模型仅在数学数据上进行训练，然后在数学和代码上进行评估，另一个是模型仅在代码上进行训练，并进行类似评估。如表 5 所示，模型在领域外任务中表现出强劲的性能，展示了强化学习的泛化能力。

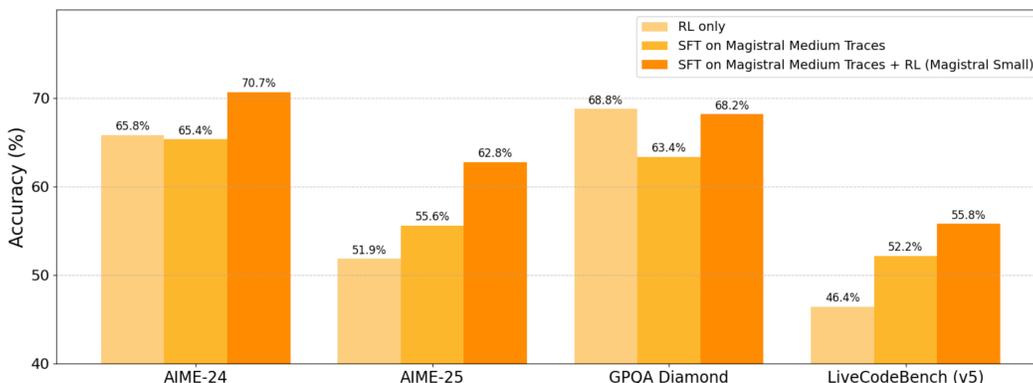
### 6.2 小模型的蒸馏与强化学习

之前的研究 [DeepSeek-AI et al., 2025] 观察到，仅依赖于强化学习的小模型可能无法达到那些从更大型推理模型中蒸馏出的模型的相似性能。然而，我们的研究结果与这种观察相矛盾：我们在 Mistral Small 3 上仅使用纯强化学习也取得了强劲的结果。如图 5 所示，我们的 Mistral Small 3 仅使用纯强化学习在 AIME’24 上表现与蒸馏版本相似。它甚至在 MATH 和

**Table 5:** 在仅数学和仅代码的 RL 过程中，跨领域泛化适用于一个 24B 模型

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

GPQA 上超过了蒸馏版本，但在 LiveCodeBench 等代码基准测试上表现略逊。这些结果表明，强化学习的益处不仅限于更大型的基础模型，对小型模型同样适用。此外，我们的发现表明，对蒸馏检查点应用强化学习可以得到更好的性能，在各种基准测试中提高了超过 5 分。



**Figure 5:** Magistral Small 在各种基准上的表现与不同的训练设置相比。我们报告了三个不同 24B 模型的表现：从零开始用 RL 训练的 Mistral Small 24B (仅 RL)，在 Magistral Medium 上推理痕迹微调的 Mistral Small 24B，以及在 Magistral Medium 痕迹上微调并随后用 RL 增强的 Mistral Small 24B，即最终的 Magistral Small。我们观察到，结合推理痕迹微调 and RL 能够获得最佳性能。

### 6.3 批处理和小批处理大小

强化学习 (RL) 算法如 PPO 或 GRPO 引入了两个不同的批次规模。批次大小，记作  $n_{batch}$ ，指的是在更新生成器权重之前收集的序列数。小批次大小  $n_{minibatch}$  表示用于计算梯度和执行单次优化步骤的序列数。需要注意的是， $n_{minibatch}$  必须能整除  $n_{batch}$ 。此外，在异步 RL 管道中，引入了第三种规模：并发序列的数量  $n_{async}$ ，代表并行生成的序列数。如果同时生成的序列数  $n_{async}$  远大于批次大小  $n_{batch}$ ，典型的序列是由  $n_{async}/n_{batch}$  个不同的策略生成的，并且可能过于偏离策略。当我们在一个批次内进行多于一次的小批次更新时，效果会更糟。

为验证此假设，我们从 Minstral 3B 开始使用 SFT 构建了一个强 3B 模型，然后在仅包含数学的数据上使用 GRPO 训练模型，训练过程中使用恒定的学习率、固定的  $n_{async} = 4096$ ，并在  $\{1024, 2048, 4096, 8192\}$  中使用不同的  $n_{batch}$  和  $n_{minibatch}$  值。

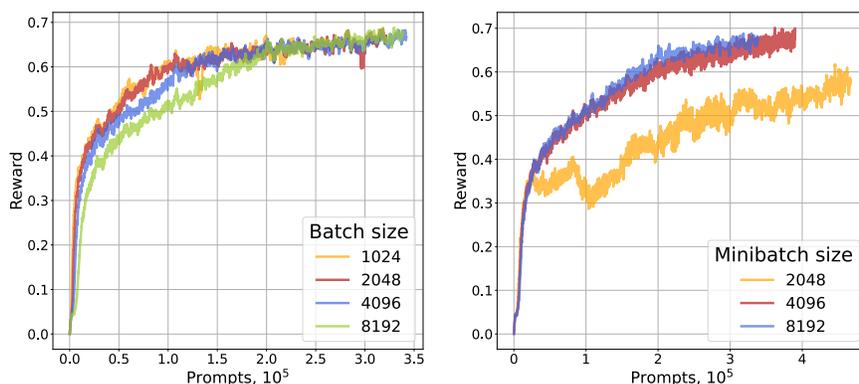
我们观察到，只要保持  $n_{batch} = n_{minibatch}$  不变，并且  $n_{batch}$  足够大，当根据处理过的提示数量绘制图表时，性能非常相似，正如图 6 (a) 所示。另一方面，当在保持  $n_{batch}$  不变的情况下减少  $n_{minibatch}$  时，性能突然下降，即使与减少到相同  $n_{minibatch}$  的  $n_{batch}$  相比，如图 6 (b) 所强调。当  $n_{batch} \leq 1024$  时，训练变得不太稳定，因此我们选择在最终训练和进一步消融过程中保持  $n_{async}/n_{batch} \leq 2$  与  $n_{batch} = n_{minibatch}$  的比例。

### 6.4 优势归一化

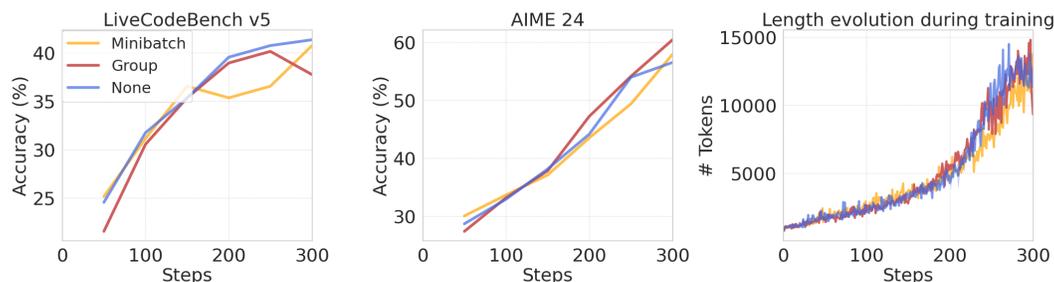
我们尝试了以下优势归一化方法：

- 小批量——在小批量内对优势进行归一化
- 组归一化 - 在一个组内相对于单个提示规范化优势
- 无归一化 - 不对优势进行归一化

先前的工作 [Liu et al., 2025, Andrychowicz et al., 2020] 指出，对于给定问题在一组生成上的归一化可能导致偏差，因为简单问题或困难问题由于其较低的标准差值而被上调权重。然而，我们没有观察到对评估性能或长度增长的任何显著影响，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。



**Figure 6:** 批量大小和小批量大小对强化学习训练奖励的影响。(a) 对于不同批量大小，在数学数据上对 3B 模型进行强化学习训练时的奖励，同时保持小批量大小等于批量大小。并生成序列的数量保持恒定定为 4096。(b) 在相同设置下，对于不同小批量大小的情况下，在固定批量大小 8192 序列时的奖励。我们观察到性能不强烈依赖于批量大小，但当一批次中有超过 2 个小批量时，性能会下降。



**Figure 7:** 在 GRPO 中使用不同优点归一化进行训练的结果。我们观察到不同的归一化方法无论是在评估性能还是在训练过程中的长度增长方面都没有导致显著差异。

## 7 分析

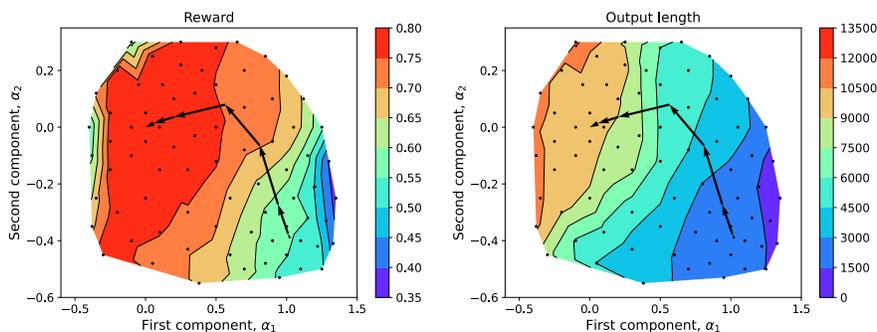
在本节中，我们研究了强化学习训练的动态性，并提供了证据表明，增加完成长度是提升模型性能的主要资源。这些动态性不会破坏之前的能力，推理能力甚至可以推广：多模态推理得到免费提升，函数调用和指令遵循保持不变甚至略有提高。此外，我们讨论了两个对我们不起作用的想法——在代码任务中根据测试完成率给予更细粒度的奖励，以及通过损失中的熵奖励项控制熵。

### 7.1 强化学习在低维空间中移动权重

为了更好地理解 Magistral 在强化学习训练中的动态，我们遵循 [Li et al., 2018] 的方法来分析仅使用 RL 的 Magistral Small 运行，并可视化最终检查点周围的损失景观。

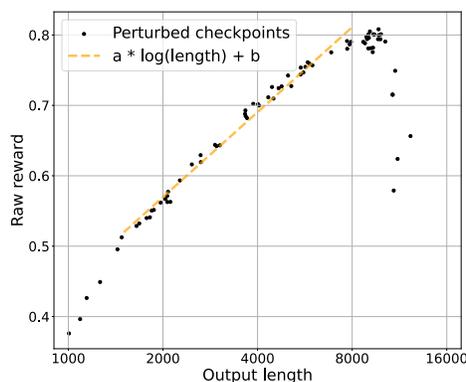
首先，我们将所有中间检查点的权重堆叠在一个矩阵  $X \in \mathbb{R}^{T \times W}$  中，其中  $T$  是检查点的数量， $W$  是权重的数量。然后，我们减去  $T$  个检查点的平均权重，并进行 PCA 分析，以在权重空间中找到矩阵  $X$  的两个主成分。由于权重空间的维度非常高，我们使用迭代 Lanczos-Arnoldi 算法 [Saad, 2003] 来找到  $X^T X$  的前两个特征向量。因此，我们得到两个分量  $c_1$  和  $c_2$ ，并对其进行 L2 归一化以使其具有单位范数。

其次，我们通过添加两个组件来扰动最终的检查点权重  $w^* \in \mathbb{R}^W$ 。我们在 512 个固定批次的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 运行中相同的奖励设置。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  $(\alpha_1, \alpha_2)$  坐标中绘制出来。



**Figure 8:** 奖励和在  $w(\alpha_1, \alpha_2)$  超平面中的长度演变。黑色箭头轨迹是在超平面上对中间检查点的 Magistral Small RL 仅运行的投影。黑色点是使用方程 ?? 计算的扰动检查点。中间值通过三角网格上的线性插值计算。

我们清楚地观察到存在一个“长度”方向——随着模型从图 8 中从右向左移动，平均奖励和输出长度增长，直到长度开始受到长度惩罚和最大允许完成长度的限制。我们额外绘制了输出长度对没有长度惩罚的原始奖励的依赖关系，观察到在图 9 中普遍存在的对数缩放。



**Figure 9:** 奖励随输出长度进行缩放。每个点对应一个通过公式 ?? 计算的扰动检查点。我们使用该检查点生成 8192 个完成项，并评估平均输出长度和原始奖励（没有长度惩罚的奖励）。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。

## 7.2 享用多模态的免费午餐

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，并带有相关的视觉编码器。在强化学习训练阶段，由于模型是在纯文本数据上进行训练的，人们可能会认为多模态性能会下降。然而，相反，我们发现这些模型不仅保留了它们的多模态能力，反而意外地发展出增强的多模态推理能力。生成的模型还展示了在多模态基准测试上的改进性能。

我们报告了用于评估推理能力的多模态基准测试结果，特别是 MathVista [Lu et al., 2024]、MMMU [Yue et al., 2024] 和 MMMU-Pro [Yue et al., 2025]。我们的结果如图 10 所示，没有出现大多数基准测试的性能回退，其中在 MMMU 上观察到显著改善（+5%，达到 70%），在 MMMU-Pro-Standard 上（+4.4%，达到 57.9%）以及在 MMMU-Pro-Vision 上（+12%，达到 52.1%）。尽管在需要文本推理的科学问题上取得的改进最为显著，我们观察到模型在所有类型的问题中都能迁移其扩展的思维过程（定性示例见图 14 15 16）。

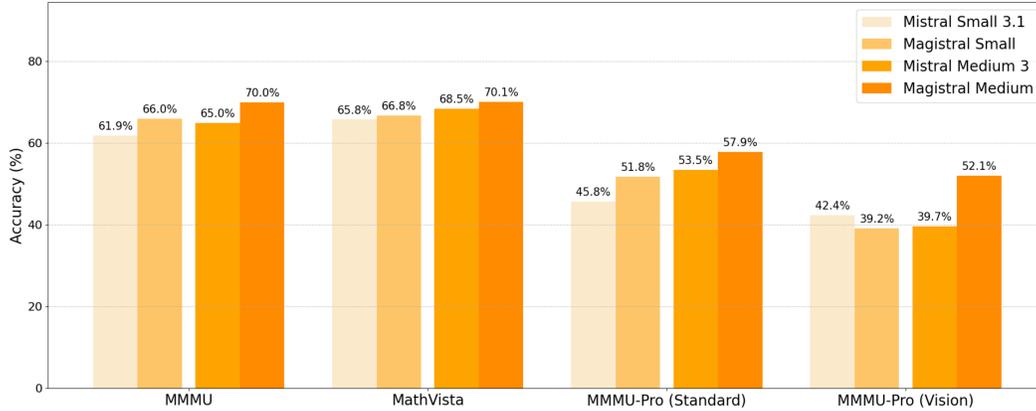


Figure 10: 在多模态基准测试中的表现。

### 7.3 强化学习对其他能力的影响

与第 7.2 节中提到的多模态功能类似，我们的 RL 检查点保持甚至提高了其调用工具和执行指令的能力 [Zhou et al., 2023] (表 6)。这使我们能够如图所示将模型即插即用与现有工具集成。

Table 6: 强化学习前后的基准测试。内部基准测试是 Mistral 的内部函数调用基准。我们使用内部版本的 IFEval，该版本修复了公共版本的一些问题。这些分数无法与其他公开共享的分数进行比较。

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

### 7.4 不成功的方法

在本节中，我们展示了我们尝试过但最终未采用的各种方法，因为它们未能带来性能提升。竞争性编程对于正确性和复杂度约束的严格要求导致奖励稀少，常常导致许多代码生成因为奖励的多样性有限而被丢弃。

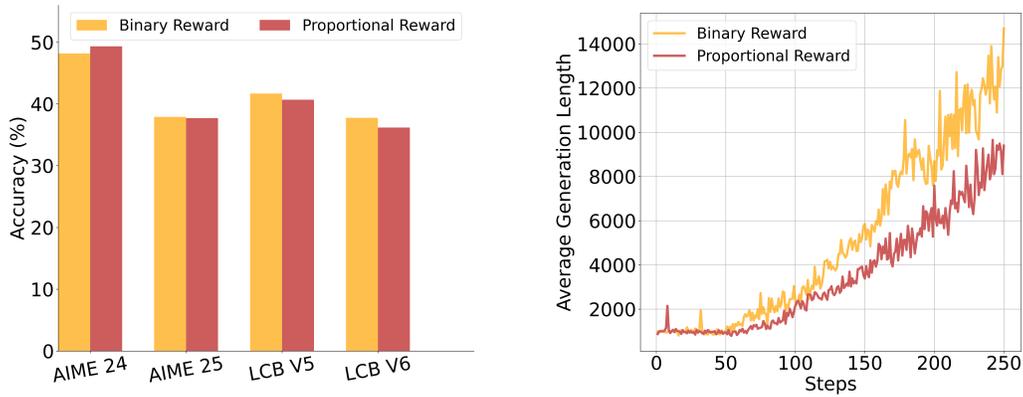
为了解决这个问题，我们尝试了比例奖励：根据通过测试的比例，而非在 2.2.2 节中讨论的二元奖励。在一个 24B 模型进行 250 步的消融实验中，我们发现使用比例奖励进行训练速度更快，丢弃的数据减少三倍。然而，这种方法导致基准测试的最终性能稍低，LiveCodeBench 中下降了 2% (图 11 a)，并且生成长度的增长较慢 (图 11 b)。

希望基于通过测试的比例给予奖励能够为强化学习训练提供比简单的通过/未通过更丰富的信号。然而，一个潜在的问题是，部分奖励可能会为错误的解决方案提供错误的信号，并对实现之间的轻微不一致更加敏感，从而可能导致不太有意义的训练批次。

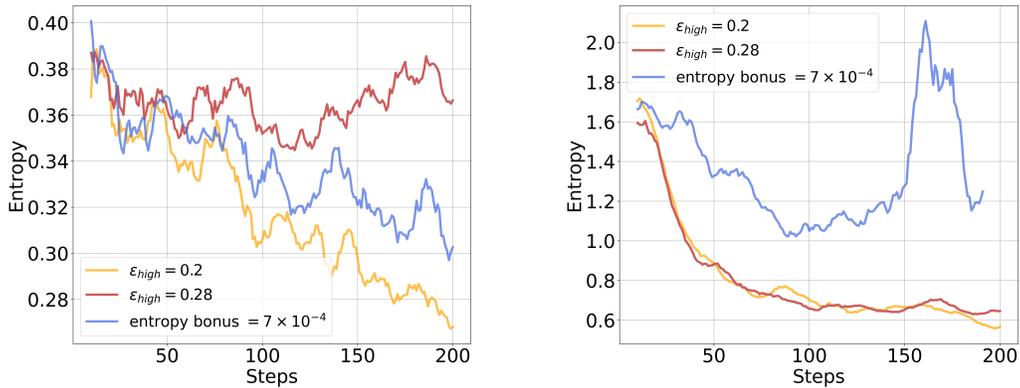
#### 7.4.1 熵目标

为了鼓励探索并防止 RL 训练过程中的熵崩溃，文献中的一种常见策略 [Schulman et al., 2017] 是添加熵奖励损失项。然而，我们发现这一策略是不稳定的，因为熵奖励的效果会根据数据集的不同而显著变化。对于仅包含数学的数据集，添加熵奖励时熵会下降，而较高的  $\epsilon_{\text{high}}$  可以维持熵，增强探索 (图 12 a)。在一个包含数学和代码的数据集中，即使使用与仅数学运行中相同的系数，熵奖励也会使熵过度增加，而较高的  $\epsilon_{\text{high}}$  可以让熵减少，从而改善利用 (图 12 b)。

相反，我们发现依赖  $\epsilon_{\text{high}}$  更为有效，这在文献中也有所提及 [Yu et al., 2025, Wang et al., 2025]。这种方法避免了与熵奖励相关的不稳定性问题。

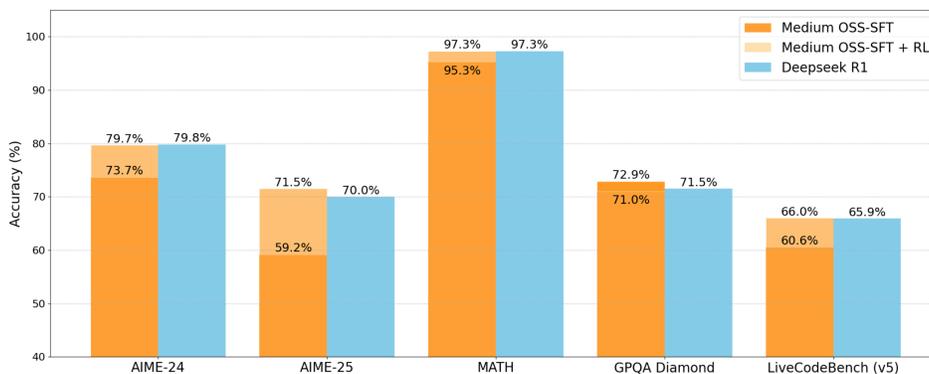


**Figure 11:** 二进制与比例奖励在代码问题中的比较。(a) 在经过 250 步训练后, AIME 和 LiveCodeBench 上的准确率, 分别使用二进制奖励和比例奖励。使用比例奖励时, 在 LiveCodeBench 上的性能低 2%。(b) 在整个训练过程中长度的变化。使用二进制奖励时长度增加更多。



**Figure 12:**  $\epsilon_{high}$  对整个训练过程中的熵分布的影响。(a) 在一个仅包括数学数据集的 3B 模型的整个训练过程中熵的变化。熵与熵奖励一起下降, 而更高的  $\epsilon_{high}$  保持熵, 从而允许更好的探索。(b) 在一个包括数学和代码数据集的 3B 模型的整个训练过程中熵的变化。即使系数与数学数据集版本相同, 熵奖励仍导致熵爆炸。较高的  $\epsilon_{high}$  表现更好, 允许熵减少。

另一种控制熵的方法是将一个 KL 项添加到 PPO 损失中。然而, 由于生成分布预期会与原始模型显著偏离, 我们发现使用 KL 惩罚主要会阻碍训练, 这与之前的发现一致 [Yu et al., 2025]。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  $\epsilon_{high}$  更为简单。



**Figure 13:** Magistral Medium 在开源轨迹上微调的基准表现。所有结果均使用 pass@1 进行报告。阴影区域突出了在监督微调基础上通过 RL 实现的额外改进。我们发现，尽管在开源轨迹上进行微调已经取得了强劲的结果，应用 RL 进一步大幅提升了性能。特别是，AIME'25 的准确率提高了超过 12%。请注意，GPQA Diamond 的性能在 RL 后从 72.9% 下降到 71.0%。

## 8 在使用 OSS 推理轨迹微调的模型上进行 RL

作为一个实验，我们还尝试了首先使用开源推理数据集 OpenThoughts [Guha et al., 2025] 和 OpenR1 [Hugging Face, 2025, Penedo et al., 2025] 的代码子集微调 Mistral Medium 3，包含这些数据集中的提示和生成，即 Deepseek R1 生成的轨迹。这包括总共约 130 万个生成。然后我们在这个微调后的检查点上使用我们最困难的数据子集运行 RL。如图 ?? 所示，应用 RL 在 SFT 检查点之上带来了显著的性能提升。值得注意的是，RL 模型在 AIME'25 上提高了 10 多个点，在 LiveCodeBench 上提高了 5 个点，在代码和数学基准测试上达到了与 Deepseek-R1 水平相当的性能。

## 9 结论

Magistral 是我们朝向具备普遍能力的强化学习系统迈出的第一步。我们期待着面前的下一个研究问题：什么样的损失和优化算法最为合适，通过模型对自身推理轨迹的自举可以解锁多少收益，或者如何扩展到下一个数量级的计算能力。展望未来，我们也对在广泛的应用中推动强化学习的边界感到兴奋，包括工具使用、集成多模态和代理。当我们探索这片新领域时，我们将继续承诺以透明和乐观的方式为科学作出贡献。

## 9.1

### 核心贡献者

阿比纳夫·拉斯托吉, 阿尔伯特·Q·姜, 安迪·洛, 加布里埃尔·贝拉达, 吉约姆·兰普尔, 杰森·鲁特, 约普·巴门特洛, 卡梅什·亚达夫, 卡尔提克·坎德尔瓦尔, 卡亚提·拉哈维·钱杜, 伦纳德·布丽尔, 露西尔·索尔涅, 马修·迪诺, 马克西姆·达林, 内哈·古普塔, 罗曼·索莱茨基, 沙加尔·维兹, 特文·莱·斯考, 易寒·王

## 9.2

### 贡献者

Adam Yang, Alexander H. 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

## References

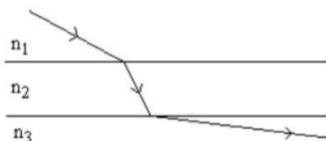
- Marcin Andrychowicz, Anton Raichuk, Piotr Staczyk, Manu Orsini, Sertan Girgin, Raphael Marinier, Léonard Hussenot, Matthieu Geist, Olivier Pietquin, Marcin Michalski, Sylvain Gelly, and Olivier Bachem. What matters in on-policy reinforcement learning? a large-scale empirical study, 2020. URL <https://arxiv.org/abs/2006.05990>.
- 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, Miaojun 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, Wanjin 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-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL <https://arxiv.org/abs/2501.12948>.
- 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, Wanjin 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, Sardia 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.
- 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 r1: A fully open reproduction of deepseek-r1, January 2025. URL <https://github.com/huggingface/open-r1>.

- 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 r1-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.
- Guilherme Penedo, Anton Lozhkov, Hynek Kydl ek, Loubna Ben Allal, Edward Beeching, Agust n Piqueres Lajar n, Quentin Gallou dec, Nathan Habib, Lewis Tunstall, and Leandro von Werra. Codeforces cots. <https://huggingface.co/datasets/open-r1/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 D rani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*, 2024.
- Youcef 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>.
- 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.
- Qiyong 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 Wenhu 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>.

Problem:



A beam 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.  $v_3 > v_1 > v_2$
- B.  $v_1 > v_2 > v_3$
- C.  $v_1 > v_3 < v_2$
- D.  $v_2 > v_3 > v_1$

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.  $v_3 > v_1 > v_2$  B.  $v_1 > v_2 > v_3$  C.  $v_1 > v_3 < v_2$  D.  $v_2 > v_3 > v_1$  The correct option that matches our conclusion is A.

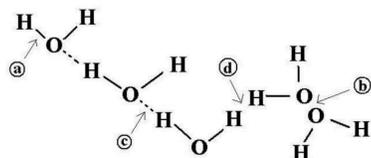
</think>

Final Answer: A

Figure 14: 由 Magistral Medium 生成的物理多模态问题及其解决方案。

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:

- 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根键，这也是一个共价键。
- 箭头 c 指向一个氢原子和另一水分子的氧原子之间的键。这看起来像是一种氢键，因为它在不同的水分子之间。
- 箭头 d 指向单个水分子内氢和氧之间的键，因此那是另一个共价键。

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: 由 Magistral Medium 生成的一种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 (watermould) 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 (watermould) 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: 一个生物学多模态问题及其由 Magistral Medium 生成的解决方案。