

穿越山谷： 小型语言模型有效长 CoT 训练之路

Renjie Luo Jiaxi Li Chen Huang Wei Lu

StatNLP Research Group, Singapore University of Technology and Design

renjie.luo@outlook.com, luwei@sutd.edu.sg

Abstract

长链式思维 (CoT) 监督已经成为增强语言模型推理能力的一种常见策略。虽然对于大型模型效果明显,但我们发现了一种现象,称为长链式思维退化,具体表现为训练在有限长链思维数据上的小型语言模型 (SLMs; $\leq 3B$ 参数) 出现显著性能下降。通过在 Qwen2.5、LLaMA3 和 Gemma3 系列上的广泛实验,我们证明这种退化在 SLMs 中普遍存在。在某些设置中,仅训练于 8k 个长链思维样例的模型在微调前失去其原始性能的高达 75%。令人惊讶的是,我们进一步观察到对于某些特别小的模型,即使训练于 220k 个长链思维样例,其微调前的原始性能也未能恢复或超越。我们的分析将这种现象归因于错误积累:尽管较长的响应增加了多步推理的能力,但同时也加剧了错误累积的风险。此外,我们发现长链式思维退化可能会对下游强化学习 (RL) 产生负面影响,虽然可以通过足够规模的监督微调 (SFT) 得到缓解。我们的研究结果挑战了关于长链思维训练对小型语言模型好处的常见假设,并为构建更有效的小规模推理模型提供了实际指导。

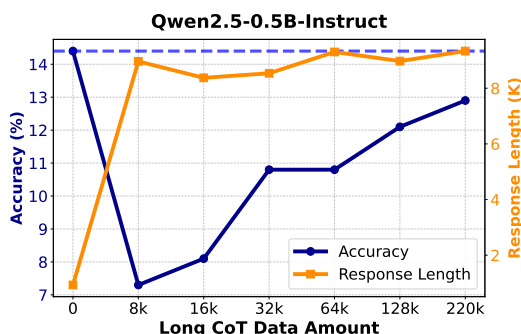


Figure 1: 在不同数量的长 CoT SFT 数据中, Qwen2.5-0.5B 的准确性和响应长度表现出显著变化。当数据量较小 (8k-16k) 时,性能显著下降,即使响应长度显著增加,这表明模型在生成更长但准确性较低的推理路径时出现了严重的失败模式。我们将这种现象称为长 CoT 退化。

1 介绍

大型推理模型,如 OpenAI-o3 (OpenAI, 2025)、Kimi-k1.5 (Team et al., 2025) 和 DeepSeek-R1 (Guo et al., 2025) 最近在复杂推理任务中展示了出色的能力。这些模型的一个关键优势是其生成思维链 (CoT) 响应的能力,这通常体现了高级、反思的推理行为。这些详细的推理响应,被称为长 CoT,是增强大型语言模型 (LLMs) 推理能力的宝贵资源。尽管在 LLM 训练中越来越多地使用长 CoT 数据,但其整合尚无一致的策略。目前的做法在模型和目标之间差异很大。例如,一些研究表明,即使使用相对少量的长 CoT 数据 (例如 $\leq 10k$ 个示例) 进行监督微调 (SFT) 也能有效增强 LLM 的推理能力 (Muennighoff et al., 2025; Xu et al., 2025; Ye et al., 2025)。其他人将有限或广泛的长 CoT SFT 与后续的 RL 训练结合起来 (Face, 2025; Chen et al., 2025; Bercovich et al., 2025; Wen et al., 2025; Guo et al., 2025)。然而,在几乎所有这些情况下,数据规模的选择往往是启发式的,目前,对长 CoT 数据规模如何影响不同训练范式和不同模型规模的模型性能的实证理解有限。

鉴于对开发和部署小型语言模型 (SLMs) 的日益兴趣,这一未充分探索的话题需要更深入的研究尤为重要,其中强大的推理能力由于其日益流行和普及至关重要 (Li et al., 2023; Hui et al., 2024; Lee et al., 2024; Agarwal et al., 2024)。与大型模型相比,SLMs 通常具有有限的容量,这可能会影响它们从冗长的监督 (如长 CoT 数据) 中进行泛化的能力 (Feng et al., 2024)。尽管长版 CoT 对大模型有效,其冗长可能会使小模型不堪重负,因此不清楚 SLMs 能够从这种训练中受益到何种程度。近期的研究甚至表明,短版 CoT 对于 SLMs 可能更为有效 (Li et al., 2025a)。然而,这些发现是基于相对较小规模的数据集 (例如,8k 个示例) 得出的,因此尚不确定是否在更大数据规模下仍然成立。

在本文中,我们对长篇 CoT 数据的规模如

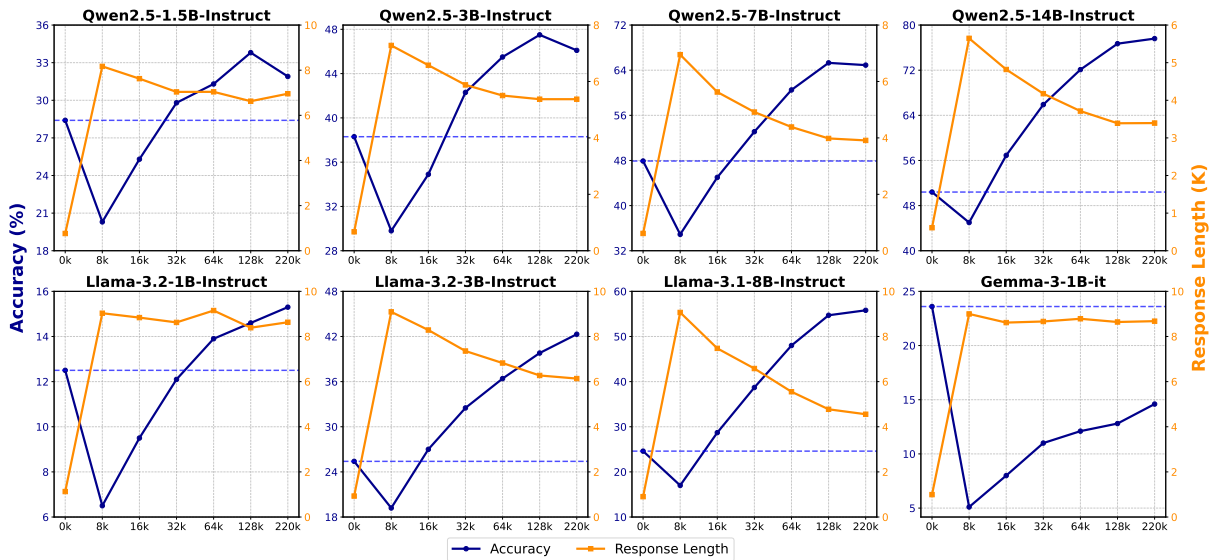


Figure 2: 对使用不同数量的长 CoT 数据训练的多个模型进行综合评估。准确性在 AIME24、AMC23 和 MATH500 上取平均值，而响应长度则是 MATH500 中 4,000 个响应对应的平均标记数。每项基准测试的结果详见附录 B.5。

何影响 SLM 的性能进行了系统的调查。我们的结果表明，仅用少量长篇 CoT 监督（例如，8k 到 16k 样本）训练的 SLM 经常遭受显著的性能退化（例如，图 1），这一现象我们称之为长篇 CoT 退化。基于这一观察，我们探讨了三个核心研究问题：1) 长篇 CoT 退化有多普遍，SLM 能否从中恢复？2) 驱动 SLM 退化的潜在机制是什么？3) 长篇 CoT 退化是否会延续到后续的强化学习 (RL) 阶段，将长篇 CoT SFT 与 RL 结合是否能减轻其影响并提高 SLM 整体训练效率？

为此，我们在各种模型规模和家族中进行广泛的实验，范围从 0.5B 到 14B，确认了长 CoT 退化的普遍性。接下来，我们围绕其潜在原因提出假设，并通过全面的实验验证这些假设。最后，我们通过分析三个代表性 SLM 在 RL 后的性能，调查了长 CoT SFT 对 RL 阶段的影响。

我们的贡献主要有三个方面：

- **Long CoT Degradation** 的经验性发现：据我们所知，我们是首次识别并系统地表征长 CoT 降解现象的研究者。这种现象在当 SLMs 用有限的长 CoT 监督进行训练时出现。我们的研究表明，这种降解在各种模型家族和广泛的 SLM 规模中持续发生，揭示了现有训练实践中的一个关键限制。（§2）
- **潜在机制分析**：我们将长 CoT 退化归因于推理输出中的长度膨胀导致的错误积累。我们的实验揭示了缺乏充分的长 CoT 监督如何导致不成比例的冗长且容易出错的响应，

最终损害了性能。（§3）

- **朝向更好的训练流程**：我们研究了长 CoT 的 SFT 如何影响 SLM 的后续 RL。我们的结果显示，虽然有限的长 CoT 曝光可能会阻碍 RL 性能，但在 SFT 期间充足的监督扩展可以显著提高 RL 的效率和最终性能，即使模型已经退化。（§4）

2 长链共轭体降解

本节对不同大小和类型的模型在长 CoT SFT 中的响应进行了全面的实证研究。我们揭示了长 CoT 监督导致的一个一致退化现象，并分析了模型在何种条件下可以恢复。此外，我们研究了长时间 CoT 持续暴露如何影响不同大小模型输出的令牌效率。

在这项工作中，我们将“长 CoT”定义为一种推理过程，其中包含显著更长的序列¹，这些序列明确包含反思、验证和子问题分解等步骤。这类链通常由大规模推理模型生成。

2.1 实验设置

数据集。 我们利用了 OpenR1-Math-220k 数据集²，其中包含来自 NuminaMath 1.5 (Li et al., 2024) 的大约 225,000 个英语数学问题，每个问题都配有由 DeepSeek-R1 生成的两到四个经过验证的推理路径。为了训练，我们为

¹例如，DeepSeek-R1-Zero 中 CoT 长回复的平均长度接近 10k 个标记，而其基础模型的平均长度不到 1k 个标记 (Guo et al., 2025)。

²<https://huggingface.co/datasets/open-r1/OpenR1-Math-220k>，依据 Apache 2.0 授权。

每个问题抽取一个正确的路径，并排除超过 16,384 个标记的离群值 (<1 % 的数据)。为了分析性能扩展，我们构建了六个逐渐增大的数据集子集：8k、16k、32k、64k、128k 和 220k 样本。

我们的研究考虑了来自 Qwen、LLaMA 和 Gemma 模型系列的九个指令微调模型，包括 Qwen-2.5 (0.5B, 1.5B, 3B, 7B, 14B)、LLaMA (3.2-1B, 3.2-3B, 3.1-8B) 和 Gemma-3-1B-it。

每个模型都使用全参数 SFT 在每个子集上进行了微调，相关一致的超参数详见附录 B。我们使用 LLaMA-Factory 框架进行训练 (Zheng et al., 2024)。

评价设置。 我们评估了三个流行的数学推理基准：AIME24³、AMC23⁴ 和 MATH500 (Hendrycks et al., 2021)。生成长度限制为 16,384 个标记。按照 Guo et al. (2025) 的评估协议，我们使用温度为 0.6 和 top-p 为 0.95 的采样方法为每个问题生成 k 个响应 (AIME24 和 AMC23 为 $k = 8$ ，MATH500 为 $k = 4$)⁵。对于每个基准，我们报告在 k 个响应上计算的平均准确率 (avg@k)。

对于所有模型系列和大小，在暴露于长 CoT SFT 之后，我们观察到准确性显著下降 (图 1 和 2)。值得注意的是，仅在 8k 长 CoT 样本训练后，Gemma3-1B-it 的准确性下降到其基线的约 25 %。即使是我们研究中最大的模型，Qwen2.5-14B-Instruct，准确性也从 50 % 下降到 45 %。此外，这种退化始终伴随着响应长度的急剧增加。这表明在模型如何处理长 CoT 监督方面存在更深层次的问题，我们在第 3 节进一步分析了这一点。鉴于在 8k 长 CoT 样本训练后观察到的一致性能下降，我们研究了随着训练样本数量增加，模型是否能够恢复，以及恢复动态如何随模型大小变化。图 2 显示，较大的模型，如 Qwen2.5-7B 和 14B，恢复更快，所需的额外样本更少，最终超过其基线性能。例如，Qwen2.5-14B 在仅训练 16k 样本后恢复并显著超过其基线，而 Qwen2.5-1.5B 在 32k 样本下略微超出其基线。对比来看，如图 1 和 2 所示，较小的模型难以完全恢复。尽管完全暴露于 220k 训练样本中，Qwen2.5-0.5B 和 Gemma-3-1B 未能达到其原始基线，最终准确性分别从 14 % 降至 11 %，从 24 % 降至 15 %。这些结果强调了从长 CoT 退化中恢复是数据

和容量依赖的。

随着长 CoT 数据的增加和模型的增大，令牌效率提高。图 2 还揭示了另一个有趣的观察点，即增加长 CoT 训练样本的数量可以提高所有模型的准确性并减少响应长度，显示出整体上的令牌效率提升。我们怀疑这是因为模型最初会模仿长 CoT 轨迹中的表面模式，导致冗长的输出。经过更多训练后，它们逐渐转向捕获潜在的推理结构，从而产生更简洁和更准确的响应。这种效果在更大的模型中更为明显。例如，通过对 32k 长 CoT 数据实例进行训练，Qwen2.5-14B-Instruct 模型在仅有 4k 令牌的平均响应长度下实现了 66 % 的准确率，而其较小的对应模型 7B 仅达到 53 % 的准确率，尽管其产生的响应平均长度为 5k 令牌。这种对比表明较大的模型更能够利用长 CoT 生成简洁而准确的答案。这些发现促使我们进一步研究驱动退化和恢复的潜在机制，我们将在下节中探讨。

3 退化背后的机制

为了更好地理解长时间连续操作退化现象，我们提出了两个假设，并设计了针对性的实验来经验验证它们。

3.1 假设

我们的假设基于两条先前的研究线索。首先，最近的研究检查了大型语言模型中的 system-2 推理，揭示了多步推理和反思等现象 (Xiang et al., 2025; Yu et al., 2024; Li et al., 2025b)。虽然这些行为在冗长的 CoT 数据中很普遍，但它们对推理链长度和结构的精确影响仍不明确。其次，关于 CoT 的先前研究表明，随着推理链的增长，中间错误的累积越来越破坏最终结果的准确性 (Wu et al., 2025)。然而，错误积累的分析迄今为止主要限于短 CoT 序列和相对较小规模的模型 (例如，具有少于 10 层的 GPT-2 变体) (Wu et al., 2025)，这就留下了一个悬而未决的问题，即在较长、更真实世界的推理环境中，使用现代大型语言模型时，类似的机制是否存在。在这些洞察的基础上，我们旨在解释在我们的实证研究中观察到的劣化和恢复行为。为此，我们提出以下两个假设：

这两个假设紧密相关：第一个旨在解释为什么模型在长链思维监督下倾向于产生较长的响应，而第二个则旨在解释导致准确性下降的原因。利用这些假设作为指导，我们在以下小节中进一步进行了实证分析。

³<https://huggingface.co/datasets/AI-MO/aimo-validation-aimc>

⁴<https://huggingface.co/datasets/AI-MO/aimo-validation-aimc>

⁵为了避免在评估长输出推理模型时由于其高重复率和跨检查点的不稳定性，贪婪解码是被避免的 (Guo et al., 2025)。

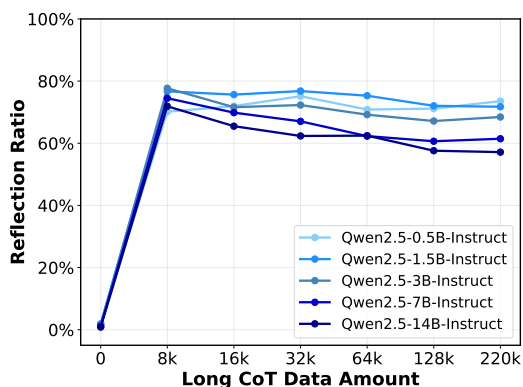


Figure 3: 不同大小的 Qwen 模型在不同数量的长 CoT 数据上训练的反射比。反射比是指在 MATH500 基准测试中展示反映行为的模型响应的比例（总共 4,000 个），并通过交叉验证识别。

3.2 反射行为分析

为了验证假设 1，我们考察模型是否在对长 CoT 数据进行微调的早期阶段迅速获取系统 2 推理的表层特征，特别是反思行为。我们的目标是确定此类模式的出现是否与第 ?? 节中观察到的响应长度增加相一致。

设置。 我们使用一种交叉验证的方法来识别模型输出中的反思行为，遵循 Liu et al. (2025)。为了稳健地检测生成响应中的自我反思，我们使用两种独立的方法：1) 基于关键字的方法，如果响应中包含任何精选的指示性关键字或短语，则将其标记为反思性的；2) 基于大型语言模型的方法，其中 GPT-4o-mini (Hurst et al., 2024) 被提示以确定响应是否表现出反思行为。该双标准策略有助于减少误报并增强可靠性。完整的实现细节和提示模板在附录中提供 B.4。

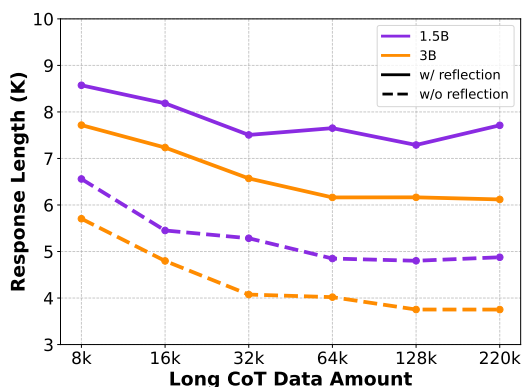


Figure 4: Qwen2.5-1.5B, 3B 指令模型在不同数量的长链式思维 (CoT) 数据上训练的平均响应长度。实线表示表现出反思行为的响应；虚线表示没有反思的响应。更多模型的结果见附录 B.6。

图 3 显示了在 Qwen 模型中反思性回应比例

的显著增加，即使只有 8k 长的 CoT 训练样本。具体来说，反思比例从低于 5% 急剧上升至大约 75%，这表明反思性行为被迅速识别并内化。此外，图 4 强调了反思性回应明显比非反思性回应更长。值得注意的是，对于 1.5B 和 3B 型号，反思性回应始终比非反思性回应多约 2,000 个标记，这一趋势在所有训练数据规模中都成立。这些发现表明，反思性行为的习得与回应长度的增长密切相关。这些发现支持假设 1：在有限的长 CoT 监督下，模型迅速采用了系统二推理的表面特征，特别是反思，反思有助于更长的回应。

3.3 累计误差分析

虽然先前的分析侧重于表层推理模式的出现，但并未直接评估输出长度如何影响答案准确性。假设 2 认为，较长的响应会增加累积错误的风险，从而降低整体准确性。在本小节中，我们在受控环境下检验该假设。

标准数学基准测试引入了诸如领域知识、问题解释和策略选择等混杂因素。这些复杂性使得难以隔离响应长度对性能的影响。为了解决这个问题，我们设计了一个合成算术基准，该基准在控制外部变量的同时保留逐步推理结构。这样我们可以直接研究输出长度如何与误差积累相关联。

我们的合成数据集中的每个实例都是由 5 到 15 个运算组成的随机生成算术表达式。为了确保难度和可解释性得到控制：

- 操作数均匀地从范围 [1, 100] 中抽取样本。
- 运算符包括加法、减法、乘法和除法。
- 所有中间结果都被限制为整数。
- 每个问题都需要固定数量的简单的、顺序的推理步骤。

这种设置通过在一致和可控条件下明确关联响应长度（算术步骤的数量）与准确性，从而能够对假设 2 进行严格的实证评估。基准中的一个示例问题如图 6 所示。

设置。 我们在 400 个合成算术问题上评估了第 2.1 节中描述的模型。按照第 2.1 节中相同的评估协议，我们使用温度为 0.6 和 top-p 为 0.95 的采样方法来生成每个问题的 $k = 4$ 个响应。我们报告了 k 个响应的平均准确率 (avg@k)。

如图 5 所示，大多数模型在训练 8k 长的 CoT 子集后，其算术准确率显著下降，同时响应长度大幅增加。例如，Qwen2.5-7B-Instruct 的准确率下降了 30%，而其平均输出长度从约 600 增加到 3600 个标记。随着更多的长 CoT 数据，性能逐渐恢复。这些趋势与在真实世界的数学基准测试（图 1 和 2）上观察到的退化和恢复

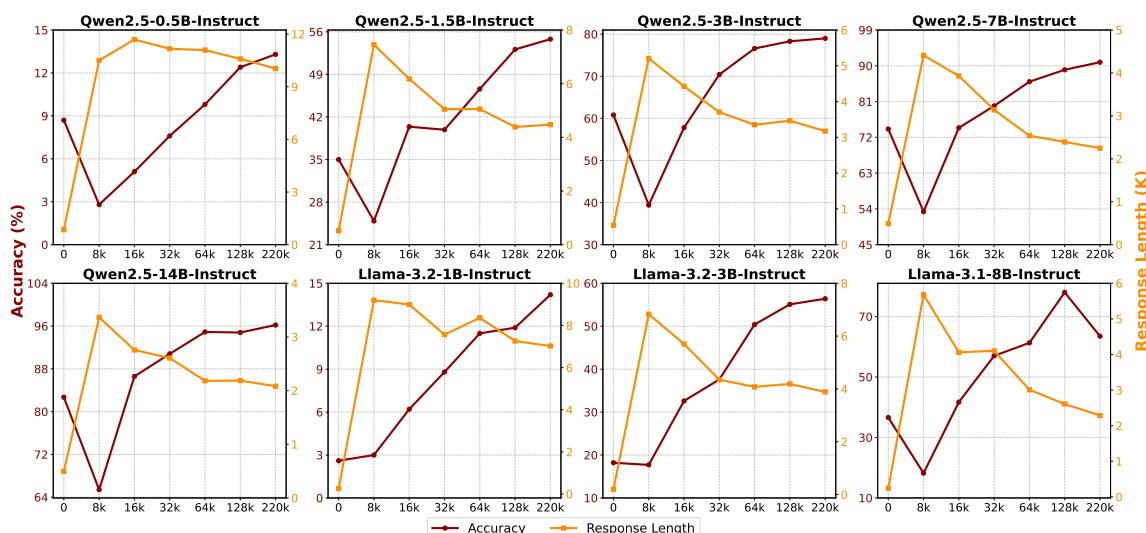


Figure 5: 在我们合成基准中，对于使用大量长 CoT 数据训练的模型，其算术精度和响应长度。大多数模型在训练 8k 子集后，算术精度急剧下降，响应长度相应增加，唯独 Llama-3.2- { 1B,3B } -Instruct 除外，该模型的初始表现已经较低 (< 20 % 精度)。

模式密切相似。定性分析进一步支持了这些发现。如附录 ?? 所示，训练于 8k CoT 数据的模型经常生成冗长的响应，带有重复的措辞（例如，“等待”）并多次重述方程，但尽管提出了替代解决方案，却仍在步骤中复合类似的算术错误。相比之下，训练于 64k CoT 数据的模型表现出更严谨的推理行为。它更有效地验证步骤并提出意图更明确的替代方案。

总体而言，这些结果为假设 2 提供了强有力的实证支持：尽管较长的回答可能使推理更加详细，但它们也增加了复合错误的可能性，导致长链推理能力下降。

4 长时间 CoT 监督对 RL 的影响

随着我们继续探索使用长 CoT 数据训练 SLMs 的不同方法的影响，我们接下来要研究的自然研究问题是，在不同设置下用长 CoT 训练后，这些 SLMs 的性能如何影响随后的 RL 阶段。

现有的方法通常分为两类：1) 在 SFT 期间使用有限数量的长 CoT 示例 ($\leq 10k$) 作为后续 RL 的冷启动 (Guo et al., 2025; Ren et al., 2025)，和 2) 对使用大量长 CoT 数据 ($> 100k$) 大幅度蒸馏的模型应用 RL (Luo et al., 2025b,a; Liu et al., 2024)。然而，对于不同水平的长 CoT 监督如何影响后续 RL 性能的系统性理解仍然未被充分探索——特别是在 SLMs 的背景下。

为了解决这一局限性，我们系统地分析了在 SFT 过程中不同量的长 CoT 监督如何影响 RL 结果，特别关注性能下降、标记效率以及整体学习动态。

4.1 训练设置

我们在三个模型上进行 RL 训练，Qwen2.5- { 0.5B, 1B, 3B } -Instruct，这些模型代表了在之前的 SFT 实验中观察到的不同退化模式的小型语言模型 (§2)。对于训练，我们采用了由 Zeng et al. (2025) 引入的数据集配置，该数据集对每个模型所使用的数据集已被实验证明是难度适中并有效的，适合其各自的规模。0.5B 模型在中等难度子集上训练，该子集由 MATH 等级 1-4 的问题组成 (Hendrycks et al., 2021)，而 1B 和 3B 模型则在困难子集上训练，其中包括 MATH 等级 3-5 的问题 (Hendrycks et al., 2021)。两个子集都包含大约 8,000 个示例。

我们使用 verl (Sheng et al., 2024) 框架进行 RL 训练。我们采用 GRPO (Shao et al., 2024) 算法，并使用基于规则的二元奖励函数：如果最终答案正确，模型会获得 1 的奖励，否则为 0。训练使用的提示批量大小为 1,024，每个提示生成 8 个回合，并将最大回合长度设置为 8,192 个标记。优化使用的微批量大小为 256。所有模型使用相同的超参数进行训练，评估设置与第 2.1 节中描述的一致。

4.2 结果

长时间的 CoT 退化对后续的 RL 训练产生负面影响。 图 7 显示，与没有任何 SFT 的基线方法相比，从用 8k 长 CoT 数据微调的检查点开始进行 RL 训练，始终会导致整个训练过程中的准确率更低且响应时间更长。此外，相对于基线的性能差距依然存在。这些结果表明，长 CoT 导致的退化难以通过 RL 缓解，并继续

Question:

Compute the answer to this arithmetic expression: $93/3+77+(12-32)-11+61*78+44*39+55$

Ground_Truth: 6606

Model's Answer (Before Training):

To compute the arithmetic expression, we will follow the order of operations (PEMDAS/BODMAS):

1. Parentheses/Brackets: Solve the expression inside the parentheses first. $12 - 32 = -20$

So, the final answer is: 6606

Number of Calculation: 10

Response Length: 502

Model's Answer (After Training):

Okay, let's see. I need to compute this arithmetic expression:

Let me compute the subtraction and addition from left to right. Starting with $108 - 20$. That's 88. Then, $88 - 11$ is 77. Then, $77 + 4758$. Wait, 77 plus 4758. Let me add those. 70 plus 4750 is 4820, and 7 plus 58 is 65, so $4820 + 65$ is 4885. So after the subtraction and addition so far, we have 4885.

So the final answer is 6601. Let me double-check my steps to make sure I didn't make a mistake.

First, 93 divided by 3 is 31. Then, $31 + 77$ is 108. Then, $12 - 32$ is -20. So the expression becomes $108 - 20 - 11 + 4758 + 1716 + 55$. Then, $108 - 20$ is 88, $88 - 11$ is 77. Then, $77 + 4758$ is 4885, $4885 + 1716$ is 6601. Yeah, that seems right. Let me verify each multiplication again.

Therefore, the answer is 6601

Number of Calculation: 40

Response Length: 1405

Figure 6: 我们合成算术基准中的一个样本问题，展示了 Qwen2.5-1.5B-Instruct 在训练 8k 长 CoT 示例前后的回答。“计算次数”表示响应中执行的算术操作总数。模型中间推理的错误在红色中突出显示。

对基于 RL 的优化产生不利影响。

如图 7 所示，在 0.5B、1.5B 和 3B 规模下使用长 CoT 数据微调的 SLMs 在 RL 训练的早期阶段显示出响应长度的快速减少，同时准确性稳步提高。这个观察结果与之前的发现一致 (Luo et al., 2025b; Liu et al., 2024)。相反，未使用长 CoT SFT 进行微调的指令基线模型在 RL 下仅显示出响应长度的轻微增加和准确性的小幅提高，这突显了这些模型进一步改进潜力的有限性 (Zeng et al., 2025)。这些结果表明，RL 可以进一步提高使用长 CoT 数据蒸馏的 SLMs 的令牌效率。值得注意的是，如第 2 节所观察到的，仅增加长 CoT SFT 数据的规模也能持续提高令牌效率，这种趋势在 RL 的作用下得到进一步增强。这一发现强调了长 CoT SFT 和 RL 在最大化 SLMs 效率方面的协同作用。

如图 7 所示，未经过长期 CoT SFT 的模型在 0.5B、1.5B 和 3B 规模的 RL 中仅表现出微小的改善。相反，从一个在 128k 长 CoT 示例上训练的检查点初始化 RL 可导致更快的准确性提升和显著更高的最终性能。具体来说，值得注意的是，对于 0.5B 型号，尽管长期 CoT SFT

最初与基线相比表现较低，但 RL 训练迅速缩小了这一差距，并进一步增强了模型的能力。值得注意的是，在 RL 后，该模型比基线提高了 13%，相对于其 RL 前状态增益超过 60%。这些结果表明，对于 SLM，即使长期 CoT SFT 暂时降低了后 SFT 性能，大规模接触长 CoT 数据在 RL 训练中仍可带来实质性的好处。

总体而言，结果表明，当使用长时间的连锁思维数据来提高小型语言模型的推理能力时，需要特别注意。有效利用这些数据首先需要在 SFT 阶段进行充分的接触。虽然仅仅经过 SFT 训练的模型可能无法获得强劲的性能，但后续的 RL 阶段可以显著增强其能力。

5 相关工作

最近在长链推理方面的进展 (Jaech et al., 2024; Guo et al., 2025) 大大提高了下游性能，特别是对于大型语言模型 (Xiang et al., 2025; Yu et al., 2024; Li et al., 2025b; Huang et al., 2024)。尽管在将长链推理知识高效蒸馏到较小模型中取得了一些进展 (Muennighoff et al., 2025; Ye et al., 2025; Yeo et al., 2025; Li et al., 2025a)，但系统且全面的评估仍然有限。

同时，RL 在推理上 (Guo et al., 2025; Schulman et al., 2017; Shao et al., 2024; Liu et al., 2025) 已经在一系列数学和编码任务中取得了可喜的结果，包括对于 SLMs (Zeng et al., 2025; Xie et al., 2025; Hu et al., 2025; Bercovich et al., 2025)。然而，对于长 CoT SFT 和 RL 训练对 SLMs 联合作用的系统性研究却很少。这个缺口促使了我们目前的研究⁶。

6 结论

在这项工作中，我们系统性地研究了长链数据的规模如何影响小型语言模型 (SLMs)。我们的研究揭示了一种一致的现象，称为长链退化，其中有限的长链监督显著削弱了不同模型系列和规模的模型性能。通过全面的实验，我们进一步分析了这种退化的机制，将其归因于由于过多冗长输出引起的错误累积——这是 SLM 特别容易受到影响的问题。最后，我们显示，虽然在数据不足时，长链 SFT 可能会阻碍后续的强化学习，但扩大长链监督可以使强化学习获得更高的效率和更好的最终性能。

我们的工作突出了当前在长 CoT 使用中进行 SLM 训练时的陷阱和误解，识别了关键的局限性，并提供了应对这些问题的实际指导。我们也希望这项工作能为未来设计有效推理模型的最佳流程的研究提供启示。

⁶由于篇幅限制，详细的综述见附录 A。

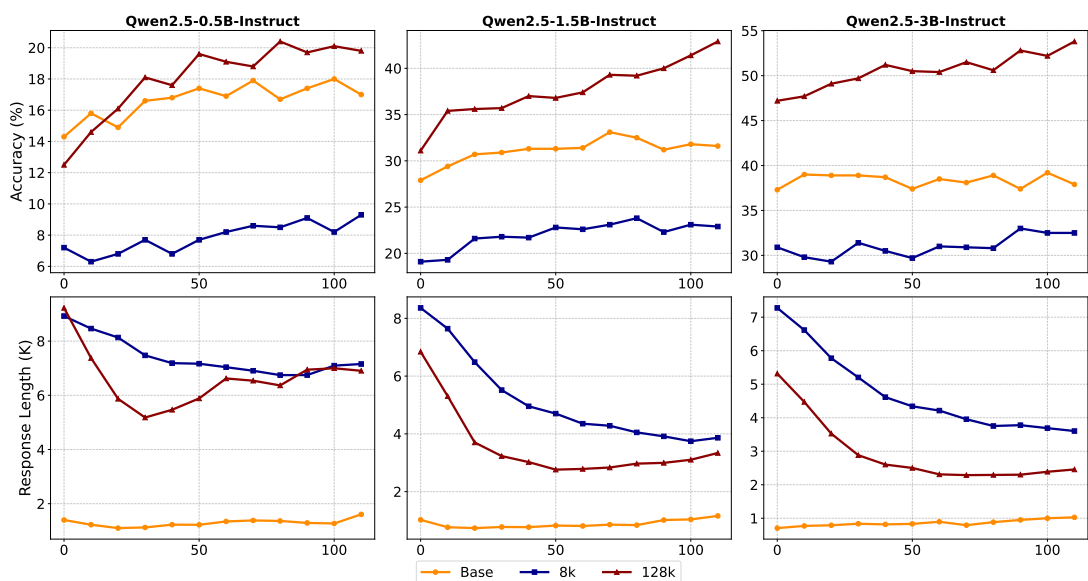


Figure 7: 长 CoT SFT 数据对 Qwen2.5 模型下游 RL 训练的影响。顶部：RL 训练模型在训练步骤中的准确性。底部：训练期间的平均反应长度。每列对应一个不同的模型规模（0.5B, 1.5B, 3B）。每条曲线代表一个 SFT 数据设置：Base（无 SFT，作为基线）、8k 和 128k（表示 SFT 过程中使用的长 CoT 例子的数量）。所有图中的横轴表示 RL 训练步骤。

7

致谢 本研究/项目由新加坡国家研究基金会在其国家大型语言模型资助计划资助（AISG 奖项编号：AISG-NMLP-2024-005），以及新加坡教育部在其学术研究基金第二级项目（MOE AcRF Tier 2 奖项编号：MOE-T2EP20122-0011）资助。本材料中表达的任何观点、发现、结论或建议仅代表作者观点，并不反映新加坡国家研究基金会或新加坡教育部的观点。

8

局限性 尽管我们的工作对长链推理监督模型（SLM）的效果提供了新的见解，我们承认有几个局限。首先，我们的分析主要关注于数学推理，这为研究长链推理监督提供了良好的结构化环境和可靠的评估标准。虽然这些发现可能对逻辑推理或代码生成等领域有帮助，但我们并没有直接研究这些领域。将我们的框架扩展到这些领域是未来研究一个很有前景的方向，尽管在这些领域获得大规模、高质量的长链推理训练数据仍然是一个实际挑战。其次，虽然我们研究了多个模型家族和规模，并观察到它们之间的持续一致模式，我们并没有明确隔离预训练数据组成的影响。先前的工作表明，预训练在塑造长链推理能力中起到重要作用。对于小型模型，预训练如何与长链推理监督相互作用的更受控调查将提供有价值的补充见解。

9

伦理声明

本研究使用公开的模型和数据集调查了长链式思维监督对小型语言模型的影响。研究不涉及人类受试者或任何敏感或专有数据。此项工作不提议或支持任何具有可预见潜在伤害或误用的应用。在实验中，我们遵守所有模型、数据和代码的许可证。

References

- Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. 2024. [On-policy distillation of language models: Learning from self-generated mistakes](#). In *Proceedings of ICLR*.
- Akhiad Bercovich, Itay Levy, Izik Golan, Mohammad Dabbah, Ran El-Yaniv, Omri Puny, Ido Galil, Zach Moshe, Tomer Ronen, Najeeb Nabwani, and 1 others. 2025. [Llama-nemotron: Efficient reasoning models](#). *arXiv preprint arXiv:2505.00949*.
- Zhipeng Chen, Yingqian Min, Beichen Zhang, Jie Chen, Jinhao Jiang, Daixuan Cheng, Wayne Xin Zhao, Zheng Liu, Xu Miao, Yang Lu, and 1 others. 2025. [An empirical study on eliciting and improving rl-like reasoning models](#). *arXiv preprint arXiv:2503.04548*.
- Hugging Face. 2025. [Open rl: A fully open reproduction of deepseek-rl](#).
- Tao Feng, Yicheng Li, Li Chenglin, Hao Chen, Fei Yu, and Yin Zhang. 2024. [Teaching small language](#)

- models reasoning through counterfactual distillation. In *Proceedings of EMNLP*.
- Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. [Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning](#). *arXiv preprint arXiv:2501.12948*.
- Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. [Measuring mathematical problem solving with the MATH dataset](#). In *Proceedings of NeurIPS Datasets and Benchmarks Track*.
- Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. [Distilling the knowledge in a neural network](#). *arXiv preprint arXiv:1503.02531*.
- Jingcheng Hu, Yinmin Zhang, Qi Han, Daxin Jiang, Xiangyu Zhang, and Heung-Yeung Shum. 2025. [Open-reasoner-zero: An open source approach to scaling up reinforcement learning on the base model](#). *arXiv preprint arXiv:2503.24290*.
- Zhen Huang, Haoyang Zou, Xuefeng Li, Yixiu Liu, Yuxiang Zheng, Ethan Chern, Shijie Xia, Yiwei Qin, Weizhe Yuan, and Pengfei Liu. 2024. [O1 replication journey-part 2: Surpassing o1-preview through simple distillation, big progress or bitter lesson?](#) *arXiv preprint arXiv:2411.16489*.
- Tingfeng Hui, Lulu Zhao, Guanting Dong, Yaqi Zhang, Hua Zhou, and Sen Su. 2024. [Smaller language models are better instruction evolvers](#). *arXiv preprint arXiv:2412.11231*.
- Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, and 1 others. 2024. [Gpt-4o system card](#). *arXiv preprint arXiv:2410.21276*.
- Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, and 1 others. 2024. [Openai o1 system card](#). *arXiv preprint arXiv:2412.16720*.
- Hojae Lee, Junho Kim, and SangKeun Lee. 2024. [Mentor-kd: Making small language models better multi-step reasoners](#). In *Proceedings of EMNLP*.
- Jia LI, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Costa Huang, Kashif Rasul, Longhui Yu, Albert Jiang, Ziju Shen, Zihan Qin, Bin Dong, Li Zhou, Yann Fleureau, Guillaume Lample, and Stanislas Polu. 2024. [Numinamath](#).
- Liunian Harold Li, Jack Hessel, Youngjae Yu, Xiang Ren, Kai-Wei Chang, and Yejin Choi. 2023. [Symbolic chain-of-thought distillation: Small models can also “think” step-by-step](#). In *Proceedings of ACL*.
- Yuetai Li, Xiang Yue, Zhangchen Xu, Fengqing Jiang, Luyao Niu, Bill Yuchen Lin, Bhaskar Ramasubramanian, and Radha Poovendran. 2025a. [Small models struggle to learn from strong reasoners](#). *arXiv preprint arXiv:2502.12143*.
- Zhong-Zhi Li, Duzhen Zhang, Ming-Liang Zhang, Jiaxin Zhang, Zengyan Liu, Yuxuan Yao, Haotian Xu, Junhao Zheng, Pei-Jie Wang, Xiuyi Chen, and 1 others. 2025b. [From system 1 to system 2: A survey of reasoning large language models](#). *arXiv preprint arXiv:2502.17419*.
- Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. 2025. [Understanding r1-zero-like training: A critical perspective](#). *arXiv preprint arXiv:2503.20783*.
- Zihan Liu, Yang Chen, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. 2024. [Acemath: Advancing frontier math reasoning with post-training and reward modeling](#). *arXiv preprint arXiv:2412.15084*.
- Michael Luo, Sijun Tan, Roy Huang, Ameen Patel, Alpay Ariyak, Qingyang Wu, Xiaoxiang Shi, Rachel Xin, Colin Cai, Maurice Weber, Ce Zhang, Li Erran Li, Raluca Ada Popa, and Ion Stoica. 2025a. [Deepcoder: A fully open-source 14b coder at o3-mini level](#). <https://pretty-radio-b75.notion.site/DeepCoder-A-Fully-Open-Source-14B-Coder-at-O3-mini-Level-1cf81902c14680b3bee5eb349a512a51>. Notion Blog.
- Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Y. Tang, Manan Roongta, Colin Cai, Jeffrey Luo, Li Erran Li, Raluca Ada Popa, and Ion Stoica. 2025b. [Deepscaler: Surpassing o1-preview with a 1.5b model by scaling rl](#). Notion Blog.
- Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. 2025. [s1: Simple test-time scaling](#). *arXiv preprint arXiv:2501.19393*.
- OpenAI. 2025. [Openai o3 and o4-mini system card](#).
- ZZ Ren, Zhihong Shao, Junxiao Song, Huajian Xin, Haocheng Wang, Wanbiao Zhao, Liyue Zhang, Zhe Fu, Qihao Zhu, Dejian Yang, and 1 others. 2025. [Deepseek-prover-v2: Advancing formal mathematical reasoning via reinforcement learning for subgoal decomposition](#). *arXiv preprint arXiv:2504.21801*.
- John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. [Proximal policy optimization algorithms](#). *arXiv preprint arXiv:1707.06347*.
- Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, and 1 others. 2024. [Deepseek-math: Pushing the limits of mathematical reasoning in open language models](#). *arXiv preprint arXiv:2402.03300*.

- Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. 2024. [Hybridflow: A flexible and efficient rlhf framework](#). *arXiv preprint arXiv:2409.19256*.
- Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, and 1 others. 2025. [Kimi k1.5: Scaling reinforcement learning with llms](#). *arXiv preprint arXiv:2501.12599*.
- Qwen Team. 2024. [Qwq: Reflect deeply on the boundaries of the unknown](#).
- Liang Wen, Yunke Cai, Fenrui Xiao, Xin He, Qi An, Zhenyu Duan, Yimin Du, Junchen Liu, Lifu Tang, Xiaowei Lv, and 1 others. 2025. [Light-rl: Curriculum sft, dpo and rl for long cot from scratch and beyond](#). *arXiv preprint arXiv:2503.10460*.
- Yuyang Wu, Yifei Wang, Tianqi Du, Stefanie Jegelka, and Yisen Wang. 2025. [When more is less: Understanding chain-of-thought length in llms](#). *arXiv preprint arXiv:2502.07266*.
- Violet Xiang, Charlie Snell, Kanishk Gandhi, Alon Albalak, Anikait Singh, Chase Blagden, Duy Phung, Rafael Rafailov, Nathan Lile, Dakota Mahan, and 1 others. 2025. [Towards system 2 reasoning in llms: Learning how to think with meta chain-of-thought](#). *arXiv preprint arXiv:2501.04682*.
- Tian Xie, Zitian Gao, Qingnan Ren, Haoming Luo, Yuqian Hong, Bryan Dai, Joey Zhou, Kai Qiu, Zhirong Wu, and Chong Luo. 2025. [Logic-rl: Unleashing llm reasoning with rule-based reinforcement learning](#). *arXiv preprint arXiv:2502.14768*.
- Haotian Xu, Xing Wu, Weinong Wang, Zhongzhi Li, Da Zheng, Boyuan Chen, Yi Hu, Shijia Kang, Jiaming Ji, Yingying Zhang, and 1 others. 2025. [Redstar: Does scaling long-cot data unlock better slow-reasoning systems?](#) *arXiv preprint arXiv:2501.11284*.
- Yixin Ye, Zhen Huang, Yang Xiao, Ethan Chern, Shijie Xia, and Pengfei Liu. 2025. [Limo: Less is more for reasoning](#). *arXiv preprint arXiv:2502.03387*.
- Edward Yeo, Yuxuan Tong, Morry Niu, Graham Neubig, and Xiang Yue. 2025. [Demystifying long chain-of-thought reasoning in llms](#). *arXiv preprint arXiv:2502.03373*.
- Ping Yu, Jing Xu, Jason Weston, and Ilia Kulikov. 2024. [Distilling system 2 into system 1](#). *arXiv preprint arXiv:2407.06023*.
- Weihao Zeng, Yuzhen Huang, Qian Liu, Wei Liu, Keqing He, Zejun Ma, and Junxian He. 2025. [Simplerl-zoo: Investigating and taming zero reinforcement learning for open base models in the wild](#). *arXiv preprint arXiv:2503.18892*.
- Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, and Zheyang Luo. 2024. [LlamaFactory: Unified efficient fine-tuning of 100+ language models](#). In *Proceedings of ACL*.

A 相关工作

A.1 长链式思维

长链推理范式首先由 OpenAI-o1 (Jaech et al., 2024) 推广, 并在 DeepSeek-R1 (Guo et al., 2025) 开源后变得广泛可用。这些 LRM 的输出以延伸和反思的长链序列为特征, 不仅展示了系统 2 认知的行为 (Xiang et al., 2025; Yu et al., 2024; Li et al., 2025b), 还为知识蒸馏提供了更丰富的中间监督信号 (Hinton et al., 2015)。因此, 下游模型性能得到了大幅提升 (Huang et al., 2024)。

后续研究集中在提高长链式思维 (CoT) 蒸馏的效率。值得注意的是, 最近的研究 (Muenighoff et al., 2025; Ye et al., 2025) 识别出紧凑但高度有效的提示子集, 表明只需 1,000 个训练示例即可实现强大的性能。然而, 这些努力主要集中在大规模模型 (例如, 具有 320 亿参数的模型) 上, 而长链式思维推理对较小模型的影响仍然很大程度上未被探索。

对小型语言模型 (8B 参数及以下) 的初步研究包括 (Yeo et al., 2025; Li et al., 2025a)。尽管这些研究为小型语言模型的长 CoT SFT 提供了宝贵的见解, 但它们基于相对有限的数据集, 该数据集由 QwQ-Preview 模型生成, 而该模型是在零-RL 训练广泛采用之前开发的 (Team, 2024)。这种限制引发了对其发现能否适用于当代训练范式的担忧。

A.2 用于推理的强化学习

DeepSeek-R1 的发布 (Guo et al., 2025) 激发了对用于推理训练的强化学习 (RL) 范式的广泛兴趣, 该范式涉及使用基于规则的奖励直接应用强化学习到基础模型, 并结合已建立的在线 RL 算法 (Schulman et al., 2017; Shao et al., 2024; Liu et al., 2025)。

RL 在具有易于验证的奖励信号的推理任务中已被证明非常有效, 特别是在数学和编码中 (Xie et al., 2025; Hu et al., 2025; Bercovich et al., 2025)。值得注意的是, 最近的研究结果表明, 即使对于小规模的语言模型, RL 仍然有效, 这强调了其广泛的适用性 (Zeng et al., 2025)。

虽然先前的工作表明, 使用短链式思维微调进行初始训练可能会限制后续增强学习在推理任务中的好处, 但最近结合广泛的长链式思维微调与增强学习的研究报告了优越的表现, 有时甚至超过了通过替代管道训练的更大模型。

然而，系统评估长链式思维微调后再进行增强学习，特别是在小型语言模型的背景下仍然有限。这一差距激发了本次研究的全面调查。

B 详细实验设置

B.1 模型

Category	Models
Qwen Family	Qwen2.5-0.5B-Instruct, Qwen2.5-1.5B-Instruct, Qwen2.5-3B-Instruct, Qwen2.5-7B-Instruct, Qwen2.5-14B-Instruct
Llama Family	Llama3.2-1B-Instruct, Llama3.2-3B-Instruct, Llama3.1-8B-Instruct
Gemma Family	Gemma3-1B-IT

Table 1: 本研究中调查的模型概述。

表 1 总结了本研究中评估的所有模型。

B.2 监督微调

监督微调 (SFT) 在数据集 $\mathcal{D} = \{(x^{(i)}, y^{(i)})\}_{i=1}^N$ 上进行，其中每个提示 $x^{(i)}$ 与一个输出 $y^{(i)}$ 配对，该输出可能包含一个长的 CoT。其目标是最大化条件对数似然 $\log p_{\theta}(y^{(i)} | x^{(i)})$ ，鼓励模型通过结构化推理再现高质量的响应。

Hyper-parameter	Long CoT Data Amount					
	8k	16k	32k	64k	128k	220k
Number of Epochs	4	4	4	3	3	2
Batch Size	8	16	32	64	128	128
Learning Rate				5×10^{-5}		
Optimizer				Adamw		
Learning Rate Scheduler				cosine		
Max Sequence Length				16384		
Warmup ratio				0.05		
Training Precision				bfloat16		

Table 2: 用于全参数监督微调的超参数。

训练设置。 我们的 SFT 训练是使用 LLaMA-Factory (Zheng et al., 2024) 在配备有 8 个 H100 (80GB) GPU 的服务器上进行的。SFT 实验总共消耗了大约 2,500 个 GPU 小时。我们为所有的 SFT 实验采用全参数微调。我们使用的详细超参数在表格 2 中展示，这些是通过初步的超参数搜索确定的。

B.3 强化学习

DeepSeek-R1 (Guo et al., 2025) 使用长期 CoT 监督微调作为冷启动进行大规模 RL，建立了一种广泛采用的面向推理的模型训练流程。在

这项工作中，我们采用相同的设置来研究在 SFT 期间的长期 CoT 监督如何影响模型在随后的 RL 阶段的行为。

对于 RL 算法，我们使用 GRPO (Shao et al., 2024)，这是一种计算上更高效的 PPO (Schulman et al., 2017) 变体，通过使用群体归一化奖励来估计优值，从而消除了单独价值模型的需要。

与 DeepSeek-R1 及类似研究一致，我们采用基于规则的二值奖励函数：如果最终答案正确，模型获得奖励 1，反之为 0。这个简单而有效的设置使我们能够孤立长 CoT SFT 对 RL 过程中优化行为的影响。

Hyper-parameter	Value
Training Algorithm	GRPO
Prompt Batch Size	1024
Rollout Per Prompt	8
Maximum Rollout Length	8192
Mini-Batch Size	256
Sampling Temperature	1.0
KL Loss Coefficient	1×10^{-4}
Learning Rate	5×10^{-7}

Table 3: 用于强化学习训练的超参数。

我们的强化学习训练使用 verl (Sheng et al., 2024) 框架进行，在一台配备八个 H100 GPU (每个 GPU 有 80GB 内存) 的服务器上运行。强化学习实验总共消耗了约 5,000 个 GPU 小时。我们实验中使用的详细超参数在表 3 中展示。

B.4 反射行为分析设置

我们采用交叉验证的方法，遵循 (Liu et al., 2025) 中提出的方法，结合关键词检测和基于大型语言模型的识别来检测模型输出中的自我反思行为。特别是，本研究使用的关键词库仅限于：重新检查、重新思考、重新评估、重新评估、再评估、重新评估、重新审查、重新检查、重新考虑、重新分析、双重检查、再次检查、再次思考、再次验证和检视步骤。

图 9 展示了使用 LLM 方法来判断一个回答是否包含自我反思行为的提示。

为了补充我们的定量分析，我们在图 13, 14, 和 15 中展示了三个具有代表性的示例。这些回答是通过不同训练设置下的 Qwen2.5-3B-Instruct 模型生成的：基线模型、在 8k 长度的 CoT 数据上微调的模型和在 64k 长度的 CoT 数据上微调的模型。基线模型以清晰简洁的推理过程正确解决了问题。相反，8k 微调的模型产生了更长且更冗长的回答。尽管提出了多种替代解决方案，但它多次犯同样的算术错误，这些错误在步骤间传播并加重了最终的错误。

回答中还表现出不流畅，包括像“等待”这样的重复短语和冗余的方程重述。64k 微调的模型显示出显著的改进：它保持了更连贯的结构，更有效地反思其步骤，并以能够导致正确解决方案的方式提出替代方案。这些示例强化了我们的假设，即较长的回答引入了更多的错误和噪声空间——除非通过对长形式推理模式的充分训练来抵消。

B.5 长协处理降解

图 11 和 12 展示了第 2 节实验的更详细的评估结果。

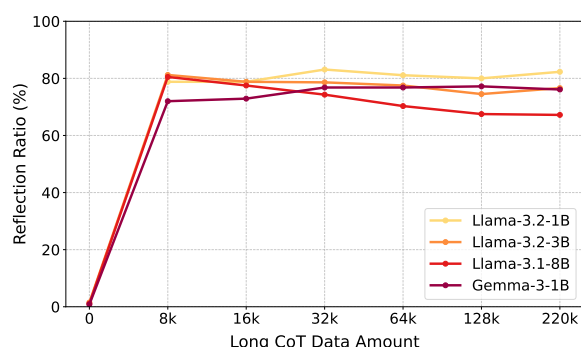


Figure 8: 在不同量的长链推理数据上训练的不同规模的 LLaMA 和 Gemma 模型的反思率。反思率是指模型对 MATH500 基准测试中 4,000 个响应中表现出反思行为的比例，这通过交叉验证辨别出来。

B.6 反射行为分析

图 8 和 10 是第 3.2 节中实验的详细评估结果。

I will send you a mathematical question along with a detailed response. Your task is to determine whether the response is attempting to answer the question. If the response is off-topic, hallucinated, random talk, or otherwise irrelevant, mark it as 0. Otherwise, assess whether the response exhibits self-reflection.

Categorization Rules:

1. **Category 0:** The response is **off-topic, nonsensical, incoherent, overly repetitive, or lacks logical reasoning.**

- Example cases:

- The response does not relate to the question.
- It contains meaningless or hallucinated content.
- It consists of excessive repetition without coherence.

2. **Category 1:** The response **attempts to answer the question** but does **not** exhibit self-reflection.

- Example cases:

- The response directly solves the problem without revisiting steps.
- No attempt is made to verify the correctness of the answer or explore alternative solutions.

3. **Category 2:** The response **demonstrates self-reflection** at any level.

- This may include:

- **Explicit self-reflection keywords**, such as: *recheck, rethink, reassess, reevaluate, re-evaluate, reevaluation, re-examine, reexamine, reconsider, reanalyze, double-check, check again, think again, verify again, go over the steps*, etc.
- **Implicit self-reflection behaviors**, such as revisiting the solution, questioning assumptions, or considering alternative approaches **without explicit keywords**.

- If any form of self-reflection is present, **always categorize it as 2**, regardless of correctness or answer quality.

4. **Category 3:** The response consists **solely of Python code for calculations** without exhibiting self-reflection.

- Example cases:

- The response only provides a Python script to compute the solution **without any verification, re-evaluation, or alternative considerations**.

Output Format:

Your response should first provide a **very brief explanation** of your analysis, followed by a **single category number (0, 1, 2, or 3)** at the end. You must include the category number at the end of your response.

Example outputs:

- ‘The response is off-topic and does not attempt to answer the question. 0.’
- ‘The response provides a direct solution without self-reflection. 1.’
- ‘The response demonstrates self-reflection. 2.’
- ‘The response consists solely of Python code without any self-reflection. 3.’

Question: {question}

Response: {response}

Figure 9: GPT-4o-mini 用于反思行为识别的提示模板。

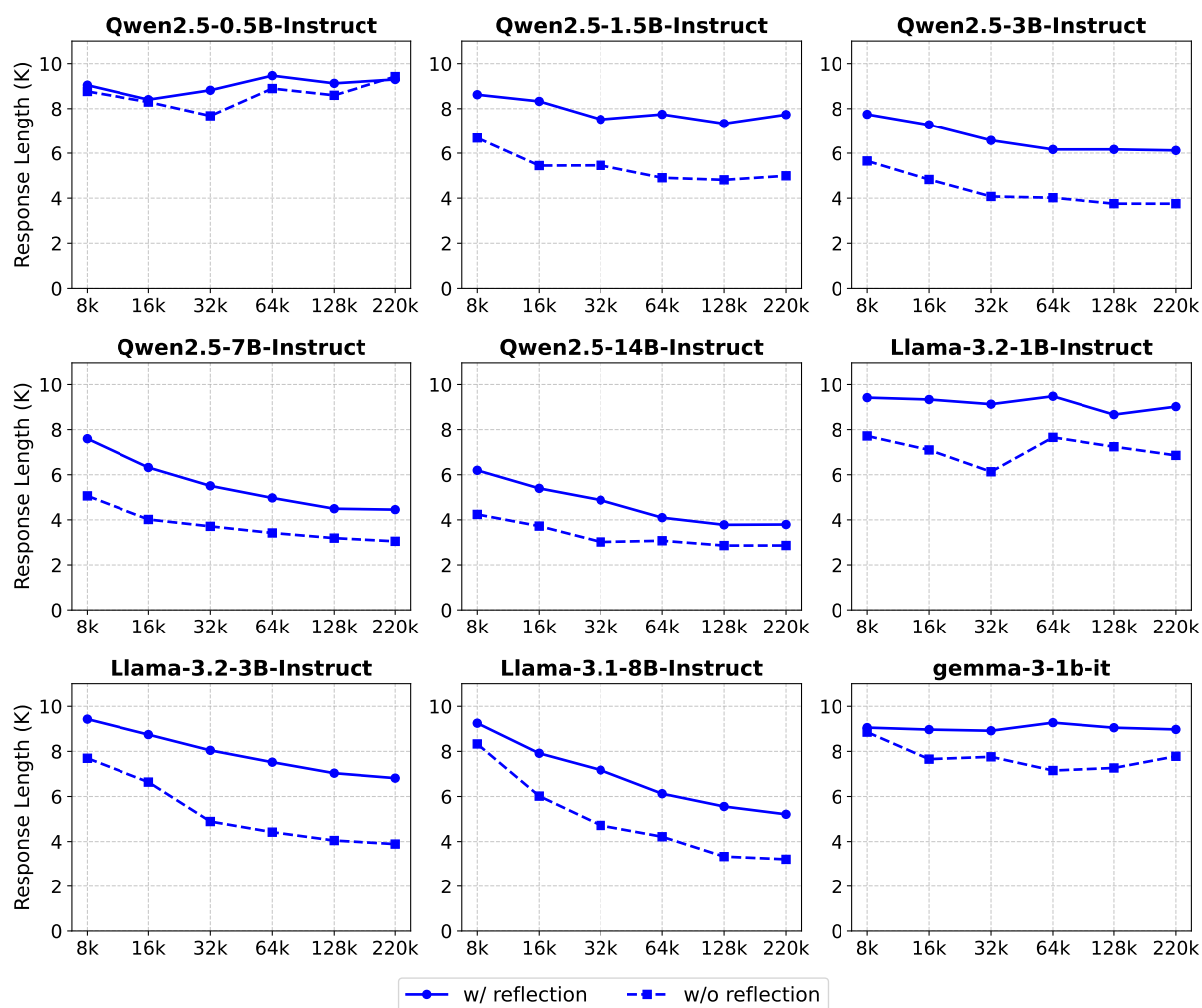


Figure 10: 多种模型在训练中使用不同数量的长 CoT 数据的平均响应长度。实线表示表现出反思行为的响应；虚线表示没有反思的响应。

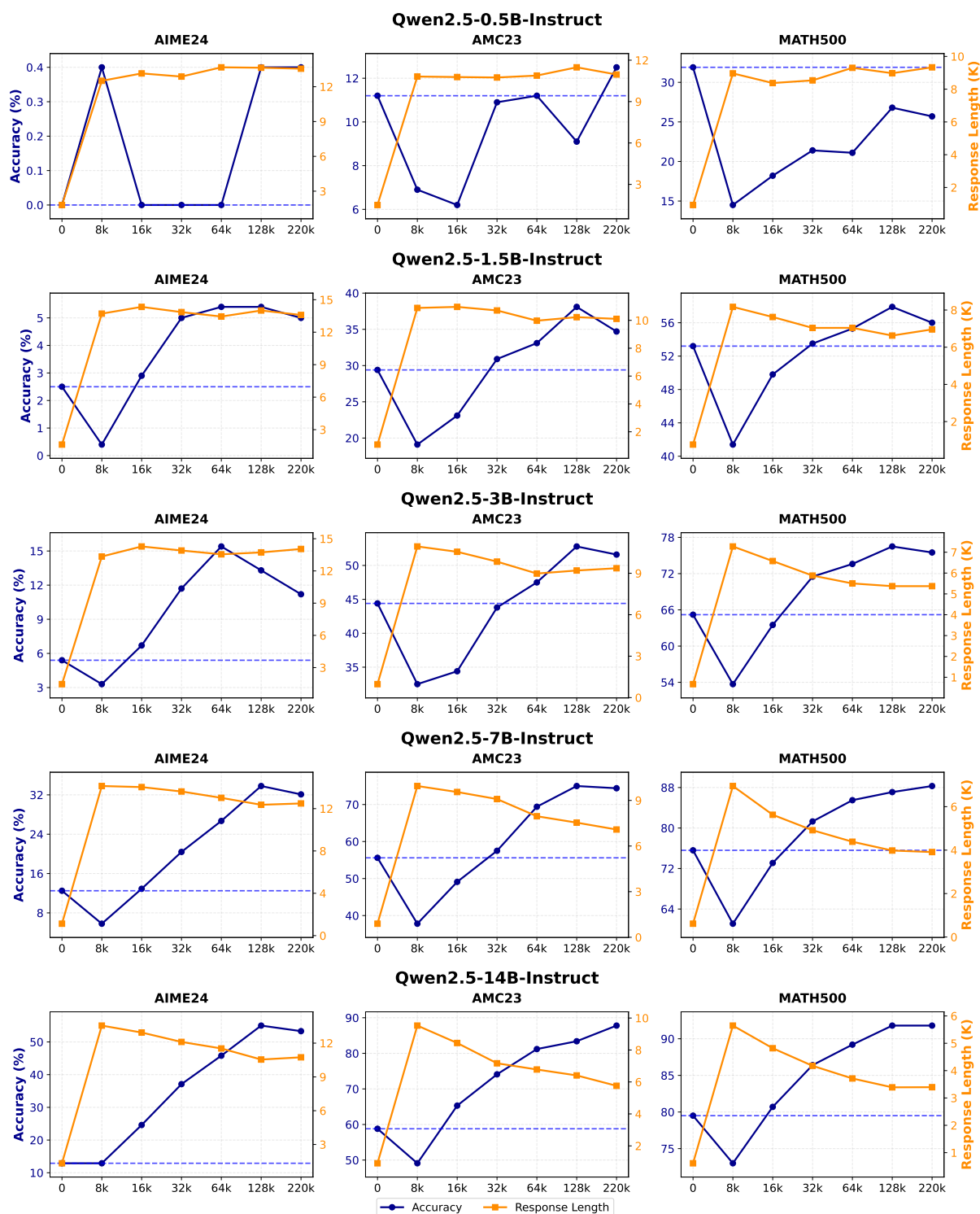


Figure 11: 对训练使用不同数量长 CoT 数据的 Qwen 模型进行全面评估。

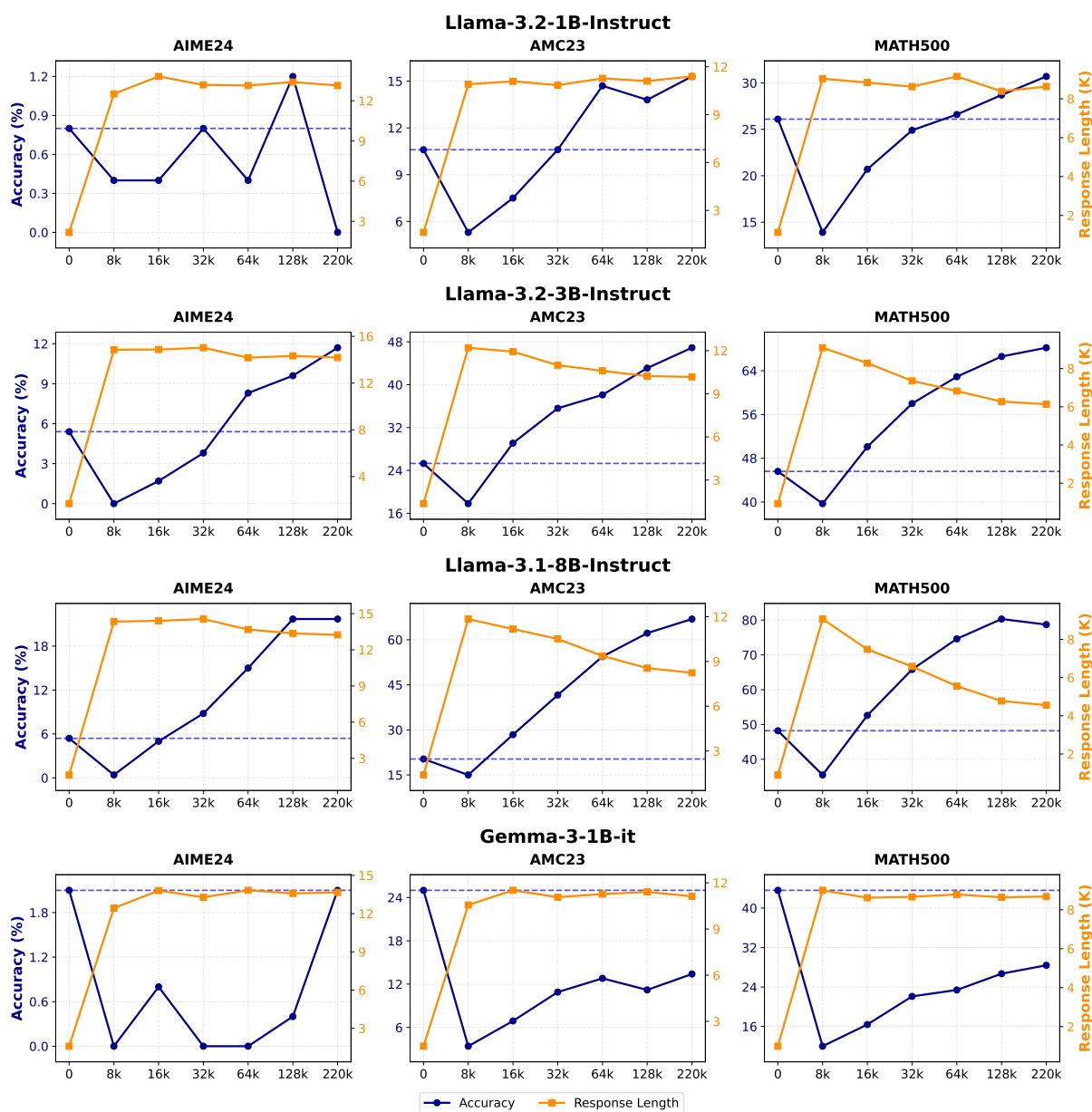


Figure 12: 全面评估使用不同数量的长链思维数据训练的 LLaMA 和 Gemma 模型。

示例 1 Qwen2.5-3B-Instruct

Question

Compute the answer to this arithmetic expression: $(14 \times 21 \times 38 + 32 + 23 + 31 + 3) - 11 + 59 + 71 + 60$

Response

To solve the arithmetic expression $(14 \times 21 \times 38 + 32 + 23 + 31 + 3) - 11 + 59 + 71 + 60$, we need to follow the order of operations, which means we should perform the multiplication before the addition and subtraction. First, calculate the product $14 \times 21 \times 38$:

$$14 \times 21 = 294$$

$$294 \times 38 = 11172$$

Now substitute this back into the expression:

$$11172 + 32 + 23 + 31 + 3 - 11 + 59 + 71 + 60$$

Next, add the numbers inside the parentheses:

$$11172 + 32 + 23 + 31 + 3 = 11261$$

Now, substitute this back into the expression:

$$11261 - 11 + 59 + 71 + 60$$

Perform the subtraction and addition from left to right:

$$11261 - 11 = 11250$$

$$11250 + 59 = 11309$$

$$11309 + 71 = 11380$$

$$11380 + 60 = 11440$$

Thus, the final answer is:

11440

Figure 13: 来自我们的合成算术基准测试的示例。响应是由 Qwen2.5-3B-Instruct 模型生成的。该模型遵循逐步方法，应用运算顺序来解决算术表达式。

示例 2 Qwen2.5-3B-Instruct 8k

Question

Compute the answer to this arithmetic expression: $(14*21*38+32+23+31+3)-11+59+71+60$

Response

Okay, let's see. I need to compute this arithmetic expression: $(14*21*38 + 32 + 23 + 31 + 3) - 11 + 59 + 71 + 60$. Hmm, that looks a bit complicated, but let's break it down step by step.

First, I should handle the operations inside the parentheses before dealing with the numbers outside. The expression starts with 14 multiplied by 21 multiplied by 38. Let me calculate that first. Multiplying those numbers might take some time, so maybe I can do them step by step. Let's see, 14 times 21. Hmm, $14*20$ is 280, and $14*1$ is 14, so $280+14=294$. Then, multiplying that result by 38. Okay, $294*38$. Let me break that down too. Maybe $294*30$ is 8820, and $294*8$ is 2352. Adding those together, $8820+2352$. Let me add them step by step: $8820 + 2000$ is 10820, then $+352$ more is 11172. So, $14*21*38$ equals 11172.

Next, the problem adds 32 to that result. So, $11172 + 32$. That's straightforward, $11172 + 30$ is 11202, plus 2 more is 11204. Then, it adds 23: $11204 + 23$. Let's see, $11204 + 20$ is 11224, plus 3 is 11227. Next, adding 31: $11227 + 31$. $11227 + 30$ is 11257, plus 1 more is 11258. Then, adding 3: $11258 + 3$. That's just 11261. So, inside the parentheses simplifies to 11261.

Now, the expression continues with minus 11: $11261 - 11$. That's easy, $11261 - 10$ is 11251, minus 1 more is 11250. Then, adding 59: $11250 + 59$. Let me check that. 11250 加 50 是 11290, 加 9 是 11299。Next, adding 71: $11299 + 70$ is 11369, plus 1 is 11370. Then, adding 60: $11370 + 60$. That's $11370 + 60$. Let me verify. $11370 + 60$: 11370 加 50 是 11320, 加上 10 是 11330。Wait, no, hold on. Wait, $11370 + 60$: $11370 + 60$. Let me do this differently. $11370 + 60$ is the same as $11370 + 6*10$. So, $11370 + 60 = 11370 + 6*10$. $11370 + 60$ is $11370 + 60$. Alternatively, $11370 + 60 = 11370 + 60$. Let me add $11370 + 60$. 11370 + 60: 11370 + 50 是 11320, 然后再 +10 是 11330。Wait, that doesn't seem right. Wait, $11370 + 60$: $11370 + 60$. Let's think of 11370 as $1137*10$. Then $1137*10 + 6*10 = 11370 + 60$. Alternatively, $11370 + 60$: $11370 + 60$. Let's do $11370 + 60$. 11370 + 50 = 11320, 然后 +10 = 11330。Wait, but $11370 + 60$ is actually $11370 + 6*10$. So, $11370 + 60 = 11370 + 60$. Let me verify with another approach. $11370 + 60$: $11370 + 60$. If I break down 60 into $50 + 10$, 然后 11370 + 50 = 11320, 然后 +10 = 11330。Alternatively, $11370 + 60$: $11370 + 60$. Let me do $11370 + 60$. Let's count from 11370: adding 10 is 11380, adding another 10 is 11390. Wait, no. Wait, $11370 + 60$: $11370 + 60$. Let me use column addition.

Wait, that's correct. So, $11370 + 60 = 11430$. So, in that case, $11370 + 60 = 11430$. So, 11430 is the sum of 11370 and 60. Okay, so that's correct. Therefore, $11261 - 11 + 59 + 71 + 60$ equals $11261 - 11 + (59 + 71 + 60)$. Let me compute $59 + 71$ first. $59 + 71$: $50 + 70$ is 120, $9 + 1$ is 10, so $120 + 10 = 130$. Then, $130 + 60$ is 190. So, $59 + 71 + 60 = 190$. Then, $11261 - 11$ is 11250, plus 190 is $11250 + 190$. Let me add those. 11250 + 190: 11250 + 100 是 11350, 然后 +90 是 11350 + 90 = 11440。So, the total is 11440.

Wait, let me check that again. $11261 - 11$ is $11261 - 10 = 11251$, minus 1 = 11250. Then, $11250 + 59 = 11250 + 59$. Let's do 11250 加 50 等于 11290, 再加 9 等于 11299。Then, $11299 + 71 = 11299 + 70 = 11369$, $+1 = 11370$. Then, $11370 + 60 = 11430$. So, total is 11430.

Alternatively, maybe there's a different way to compute the total without breaking it down so many times. Let me see. The original expression is $(14*21*38 + 32 + 23 + 31 + 3) - 11 + 59 + 71 + 60$. So, the first part is $14*21*38 + (32 + 23 + 31 + 3)$.

So, that's 11261. Then, $11261 - 11 = 11250$, $+59 = 11250 + 59 = 11250 + 50 + 9 = 11290 + 9 = 11299$. Then, $+71 = 11299 + 70 = 11369$, $+1 = 11370$. Then, $+60 = 11370 + 60 = 11430$. So, total is 11430.

Wait, maybe the error is in the initial step of adding the numbers inside the parentheses. Let me check that again. The first part is $(14*21*38 + 32 + 23 + 31 + 3)$. Let's compute each term: $14*21*38$: $14*21=294$, $294*38=11172$. Correct.

So, inside the parentheses is 11261. Then, subtract 11: $11261 - 11 = 11250$. Correct.

Then, add 59: $11250 + 59 = 11250 + 50 + 9 = 11290 + 9 = 11299$. Correct.

Add 71: $11299 + 71 = 11299 + 70 + 1 = 11369 + 1 = 11370$. Correct.

Add 60: $11370 + 60 = 11370 + 60 = 11430$. Correct.

So, the total is 11430. Therefore, the first method is correct. Then, why does the second method give a different result? Because in the second method, the user added the constants first.

Therefore, the total is 11430. Therefore, the second method is incorrect because it misapplies the order of operations. The user added the constants first, which is not the case in the original expression. The original expression requires the constants to be added after the product is computed and before the subtraction and additions. Therefore, the correct total is 11430.

Therefore, the final answer is 11430.

Figure 14: 来自我们合成算术基准的例子。响应由在 8k 长 CoT 数据上微调过的 Qwen2.5-3B-Instruct 生成。尽管提出了多种替代解决方案，该模型仍犯了类似的计算错误，这些错误在随后的步骤中传播并导致合并错误。错误步骤在 红色 中被突出显示。响应被截断，因为它超过了 5,000 个标记。

示例 3 Qwen2.5-3B-Instruct 64k

Question

Compute the answer to this arithmetic expression: $(14*21*38+32+23+31+3)-11+59+71+60$

Response

Okay, so I need to compute this arithmetic expression: $(14*21*38 + 32 + 23 + 31 + 3) - 11 + 59 + 71 + 60$. Hmm, let me break this down step by step. First, I should handle the operations inside the parentheses, which is the product of 14, 21, and 38, then add the other numbers. Let me start with that.

Calculating 14 multiplied by 21 . Let me do $14*20$ first, which is 280 , and then add $14*1$, which is 14 . So $280 + 14 = 294$. Okay, so $14*21$ is 294 . Now, I need to multiply that result by 38 . Hmm, $294*38$. Let me break that down. Maybe $294*40$ would be easier, but subtract $294*2$. Wait, $294*40$ is $11,760$ and $294*2$ is 588 . So $11,760 - 588 = 11,172$. So $14*21*38$ is $11,172$. 让我再检查一下以确保。 $294*38$: $294*30$ 是 $8,820$, $294*8$ 是 $2,352$ 。把它们加在一起: $8,820 + 2,352$ 。让我看看, $8,000 + 2,000$ 是 $10,000$, $820 + 352$ 是 $1,172$ 。所以 $10,000 + 1,172$ 是 $11,172$ 。是的, 匹配了。所以那部分是正确的。

Now, add the numbers 32 , 23 , 31 , and 3 . Let's do that step by step. $32 + 23$ is 55 . Then $55 + 31$ is 86 . Then $86 + 3$ is 89 . So the sum inside the parentheses is $11,172 + 89$. Let me compute that. $11,172 + 80$ is $11,252$, then $+9$ is $11,261$. So the parentheses part equals $11,261$.

Now the entire expression is $11,261 - 11 + 59 + 71 + 60$. Let me handle the subtraction and addition from left to right. So first, $11,261 - 11$. That's straightforward: $11,261 - 10$ is $11,251$, then subtract 1 more is $11,250$. So now we have $11,250 + 59 + 71 + 60$.

Let me add those numbers. $59 + 71$. $50 + 70$ is 120 , $9 + 1$ is 10 , so total 130 . Then $130 + 60$ is 190 . So adding 190 to $11,250$ gives $11,250 + 190 = 11,440$. 让我再验证每一步, 以确保我没有犯错。

首先, $14*21*38$: $14*21$ 是 294 , $294*38$ 是 $11,172$ 。然后加上 $32+23+31+3$: $32+23=55$, $55+31=86$, $86+3=89$ 。所以 $11,172 + 89 = 11,261$ 。然后 $11,261 - 11 = 11,250$ 。接着加上 $59+71+60$: $59+71=130$, $130+60=190$ 。 $11,250 + 190 = 11,440$ 。看起来是正确的。

Wait, let me check the addition again. $59 + 71$: $59 + 70$ is 129 , plus 1 is 130 . Then $130 + 60$ is 190 . Correct. Then $11,250 + 190$: $11,250 + 100$ is $11,350$, plus 90 is $11,440$. Yes. So the final answer should be $11,440$.

但为了更全面, 让我再一步一步地计算整个表达式, 可能使用不同的顺序或用另一种方式验证乘法。

或者, 我可以重新计算 $14*21*38$ 。我们来看, $14*21$ 是 294 , 然后 $294*38$ 。我们可以计算 $294*38$ 为 $(300 - 6)*38$ 。那么 $300*38 = 11,400$, $6*38=228$ 。所以 $11,400 - 228 = 11,172$ 。答案一样。然后加上 $32+23+31+3$: $32+23=55$, $55+31=86$, $86+3=89$ 。所以 $11,172 + 89 = 11,261$ 。然后减去 11 : $11,261 - 11 = 11,250$ 。然后加上 $59,71,60$ 。先算 $59 + 71$: $59 + 70$ 是 129 , 加 1 是 130 。然后 $130 + 60$ 是 190 。所以 $11,250 + 190$ 是 $11,440$ 。所有步骤都对。

Alternatively, maybe compute the entire expression as written:

$(14*21*38 + 32 + 23 + 31 + 3) - 11 + 59 + 71 + 60$

Let me compute all the additions first before subtracting:

First, compute the sum inside the parentheses: $14*21*38 + 32 + 23 + 31 + 3$.

But we already did that. Then subtract 11 , then add 59 , 71 , 60 .

或者, 可以逐步计算每一项:

Start with $14*21*38$. Let's compute that again. $14*21$ is 294 . Then $294*38$. Let me compute $294*38$ as $294*(30 + 8) = 294*30 + 294*8$. $294*30$ is $8,820$. $294*8$ is $2,352$. Adding those gives $8,820 + 2,352 = 11,172$. Correct.

Then $32 + 23 + 31 + 3$. Let's add them in pairs: $32 + 31 = 63$, $23 + 3 = 26$. Then $63 + 26 = 89$. Correct. So the parentheses total $11,172 + 89 = 11,261$.

Now subtract 11 : $11,261 - 11 = 11,250$. Then add 59 , 71 , 60 . Let's add $59 + 71$ first. $59 + 70$ is 129 , plus 1 is 130 . Then $130 + 60 = 190$. Then $11,250 + 190 = 11,440$. All steps confirm the same result.

或者, 可以按照写出的整个表达式进行计算:

$14*21*38 + 32 + 23 + 31 + 3 - 11 + 59 + 71 + 60$

But that's the same as before. 或者, 按顺序计算每一项:

$14*21*38 = 11,172$

.....

$+60 = 11,440$

Yes, same result. So regardless of the order, the answer is $11,440$.

我认为这很稳妥。让我检查一下我在最后的加法中是否犯了任何算术错误。 $11,250 + 190$. $11,250 + 100 = 11,350$. Then $+90$ is $11,440$. Correct.

Alternatively, 190 is $100 + 90$. $11,250 + 100$ is $11,350$. $11,350 + 90$ is $11,440$. Yep.

So I think that's the correct answer. $11,440$. So the final answer is 11440 .

Final Answer 11440

Figure 15: 来自我们合成算术基准测试的示例。该响应由 Qwen2.5-3B-Instruct 生成, 该模型在 64k 长的 CoT 数据上进行了微调。模型经常进行验证步骤, 并偶尔提出替代解决方案。验证步骤和替代解决方案路径在蓝色中进行了突出显示。