简单问题快解，复杂问题深探：通过加长惩罚实现高效推理

Zehui Ling[1,3*], **Deshu Chen**[1,2,3*], **Hongwei Zhang**[1,2,3], **Yifeng Jiao**[3], **Xin Guo**[3], **Yuan Cheng**[1,3 †]

[1]Artificial Intelligence Innovation and Incubation Institute, Fudan University
[2]School of Data Science, Fudan University
[3]Shanghai Academy of Artificial Intelligence for Science
[†]Corresponding author:cheng_yuan@fudan.edu.cn

大型语言模型最近取得了惊人的进展，特别是在推理能力方面。然而，推理能力的增强是以显著增加额外开销为代价的。例如，通过在输入提示中添加“让我们一步一步思考”来实现的链式推理（CoT），通过推理链分解问题解决的步骤，最终得到正确的答案。CoT 非常有效，结果是许多现有模型，如 DeepSeek-R1，使用 CoT 数据对模型进行强化学习，取得了惊人的成果。

Original Model (265 tokens)

First, I need to determine the total amount Keegan and Tashay have earned so far by adding their individual earnings. Keegan earned $ 83, and Tashay earned $ 91. Adding these together gives: 83 + 91 = 174 Next, I'll subtract their total earnings from the goal of $ 200 to find out how much more they need to earn: 200 - 174 = 26 Therefore, they still need to earn $ 26 to reach their goal. </think> To determine how much more Keegan and Tashay need to earn to reach their goal of $ 200, follow these steps: 1. **Calculate the total amount earned so far:** Keegan's earnings + Tashay's earnings= $ 83 + $ 91 = $ 174 2. **Determine the remaining amount needed to reach the goal:**

$$Goal - Totalearned = \$200 - \$174 = \$26$$

Therefore, they need to earn an additional $\boxed{\$26}$ to reach their goal.

Ours (54 tokens)

First, add Keegan's and Tashay's earnings: $ 83 + $ 91 = $ 174 . Subtract this from their goal: $ 200 - $ 174 = $ 26 . $\boxed{26}$

Figure 1: 在相同提示下的模型输出比较。我们的方法产生了更短但准确的响应，显示出更高效的推理能力。

尽管生成连续思考（CoT）的方法在解决复杂和困难问题时非常有效，但当应用于解决简单问题时，这种方法也会产生显著的开销 (Feng et al., 2023) 。例如，在解决像“3 + 4 = ?”这样简单的问题时，具有深度思考能力的 Deepseek-R1 首先会花费数秒钟思考，并消耗大量的标记，才得出答案。我们希望避免这种显著的延迟和资源消耗。最近，已经提出了几种方法，旨在缩短模型在回答简单问题时所需的推理路径 (Sui et al., 2025) 。提出了多种方法来实现这一目标。

(1) 提示引导的高效推理 (Han et al., 2024; Xu et al., 2025; Ding et al., 2024; Lee et al., 2025) 侧重于设计专门的提示，以引导模型生成更短、更直接的推理路径，从而高效地回答问题。一个优点是不需要额外的模型训练。然而，对于参数较小的模型，基于提示的控制往往效果较差，其可控性和性能可能不如预期。

(2) 可变长度推理链 (Xia et al., 2025; Ye et al., 2025; Ma et al., 2025; Munkhbat et al.; Liu et al., 2024) 通过对包含不同长度推理链的数据集进行监督微调来训练大型语言模型，使模型能够根据输入的复杂性调整其推理深度。其优点之一在于能够通过构建定制的数据集来微调模型，这需要较少的计算资源，并在训练数据域内实现出色的性能。然而，它在训练分布之外的例子上表现出有限的泛化能力和中等的表现。

(3) 长度奖励设计 (Team et al., 2025; Arora and Zanette, 2025; Luo et al., 2025; Qu et al., 2025) 在强化学习中塑造奖励函数，通过对简洁而正确的答案给予更高的奖励，同时惩罚过长或不正确的回答，从而鼓励模型生成较短的推理路径。这需要更高的计算资源；然而，在训练后，这赋予模型更强的泛化能力，并且在推理过程中不依赖于额外的提示或标记。

尽管这些方法都旨在减少语言模型推理链的长度，但它们通常忽视了问题难度的变化。因此，这些方法可能无意中妨碍了模型处理复杂或具有挑战性任务的能力。这引发了一个重要的研究问题：

我们能否教语言模型像人类一样思考—对简单问题反应迅速，对复杂问题深入思考？

直观上，困难问题通常需要更多的推理和解决问题的步骤 (Kahneman, 2011; Perez et al.,

2020)。为了得到正确答案，大型语言模型通常会在具有挑战性的问题上花费更多的标记量。受这一观察启发，我们提出使用模型响应长度作为问题难度的指标。我们的目标是开发一种方法，使模型在简单问题上减少资源消耗的同时，在复杂问题上保持高准确性。

我们通过适应 RLOO 算法 (Ahmadian et al., 2024) 并修改奖励函数来实现这一点。具体来说，我们在奖励结构中引入了长度惩罚机制，使得模型对简短且正确的回答获得最高奖励，对较长但仍正确的回答获得稍低的奖励，而对不正确的答案则没有奖励。这鼓励模型生成简洁且准确的推理路径，根据问题的难度调整其计算，如图 1 所示。我们观察到，即使模型仅在相对简单的 GSM8K 数据集上进行训练，它也表现出较强的推广能力，适用于更具挑战性的数学基准。我们在两个模型变体上评估了我们的方法：DeepSeek-R1-Distill-Qwen-1.5B 和 DeepSeek-R1-Distill-Qwen-7B。对于 1.5B 模型，我们的方法取得了显著的改进。在 GSM8K 数据集上，它将输出的令牌数量减少了 40 %，同时将准确率提高了 10 %。在更具挑战性的 AIME2024 数据集上，模型减少了 15 % 的令牌使用，同时准确率提高了 4.8 %。对于 7B 模型，我们的方法表现出更高的效率。在 GSM8K 上将令牌使用减少多达 90 %，但准确率仅小幅下降 1.4 %。在 AIME2024 上，当令牌数量减少 20 % 时，准确率几乎没有变化。

总之，我们的贡献可以总结如下：

- 我们提出了一种基于新颖奖励函数的方法，该方法对简单问题有效施加长度惩罚，同时对困难问题几乎不施加长度惩罚。
- 通过分析优势函数，我们证明了相比于现有的压缩思维链推理的强化学习方法，我们的方法与“简单题快速，困难题深刻”这一原则更好地对齐。
- 大量实验验证了我们方法的有效性，深入分析为该领域的未来研究提供了宝贵的见解。

1 相关工作

大型推理模型近年来，大型语言模型在处理复杂推理任务方面展现出了非凡的潜力。类似连锁思考和树状思考 (Yao et al., 2023) 这样的开创性方法，使得这些模型能够通过逐步推理过程执行多步逻辑推理。同时，诸如 OpenAI (Achiam et al., 2023)、Deepseek-R1 (Guo et al., 2025)、QwQ-preview (Team, 2024) 和 Kimi (Team et al., 2025) 等模型通过大规模强化学习进一步增强了其推理能力，使其具备如分支和验证等高级技能，从而显著提升了其总体推理性能。

高效推理尽管推理模型和连锁思考方法提升了模型的推理能力，它们也带来了更高的计算成本。为了解决这一问题，各种方法被提出以提高推理效率。Token-Budget (Han et al., 2024) 估算回答问题所需的最少 token 数量，并通过提示引导模型将推理过程限制在这个 token 范围内，从而减少计算开销。然而，该方法在估算问题难度时会使用额外的 token。TokenSkip (Xia et al., 2025) 评估响应中每个 token 的重要性，适当标记它们，并在部分不太重要 token 被移除的数据集上微调模型。这通常导致省略连词的答案，产生不太连贯的响应。另一种方法是训练语言模型进行高效推理 (Arora and Zanette, 2025)，通过在 RLOO 中添加基于动作数量 α 的长度惩罚来修改奖励函数，随后在混合数据集上进行强化学习。然而，它并没有在困难样本和简单样本之间区分长度惩罚。

2 方法论

2.1 概述

我们的目标是使模型能够进行高效的推理，即在减少标记使用的同时给出准确的答案。对于简单问题，鼓励模型生成简洁的回应。对于更具挑战性的问题，允许模型优先考虑准确性而非简洁性。为实现这一目标，我们应用了强化学习，并设计了一个精心设计的奖励函数，引导模型在效率与正确性之间达到平衡。我们的流程可以在图 Figure 2 中看到。

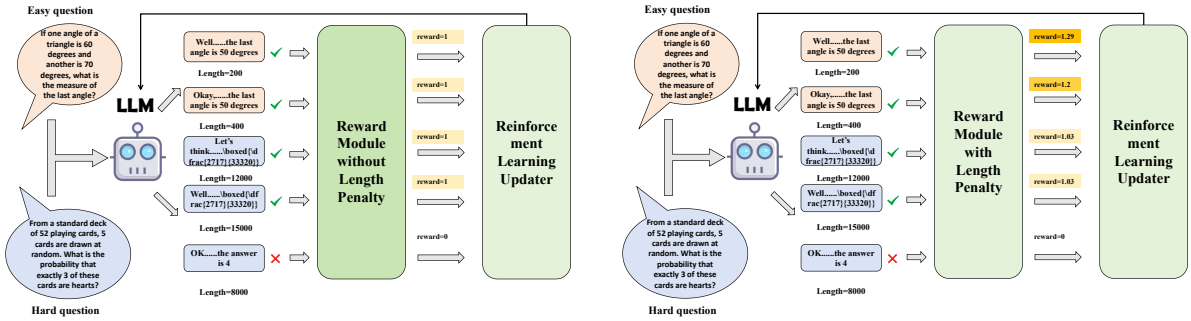
2.2 问题设定

我们考虑一个由 θ 参数化的大型语言模型，记为 π_θ。给定一个输入序列 x，模型生成一个回应 y，其中 y 是从条件分布 $\pi_\theta(\cdot \mid x)$ 中抽样的。由于我们的评估涉及具有客观、可验证答案的问题，我们定义准确性如下：

$$\text{accuracy} = \begin{cases} 1, & \text{if } y = y^*, \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

其中 y^* 表示正确答案，y 是模型生成的答案。只有最终答案需要与真实值匹配，中间推理步骤或完整输出序列无需完全相同。

在强化学习中，近端策略优化（PPO）算法 (Schulman et al., 2017) 是最广泛使用的方法之一。然而，由于 PPO 依赖于评论模型和外部奖励模型，它引入了大量的内存开销。为了避免这一限制，我们采用了更简单的 REINFORCE



(a) 无长度惩罚的 RL。

(b) 带长度惩罚的强化学习（$\alpha = 4, \gamma = 0.5$）。

Figure 2: 原始强化学习和我们方法的区别。相同的颜色表示与答案对应的问题。在我们的方法中，模型根据问题的难度调整长度惩罚：对简单任务施加高惩罚以鼓励简洁的回答，而对复杂任务则将惩罚最小化以允许更全面的回答。无论回答长度如何，不正确的输出都会获得 0 奖励。

框架，它消除了对评论模型的需求，从而显著减少了内存消耗。此外，由于我们的任务涉及具有确定性答案的数学问题，定义在方程（1）中的准确度函数可以自然地作为奖励信号。因此，我们也省去了奖励模型。

在 Reinforce 框架中，梯度更新由以下公式给出：

$$\mathbb{E}_{x\sim\mathcal{D},y\sim\pi_\theta(.|x)}\left[R(y,x)\nabla_\theta \log \pi_\theta(y|x)\right], \quad (2)$$

为了减少偏差，我们采用 RLOO 方法。我们针对每个答案生成 k 个样本，并通过减去剩余 $k-1$ 个样本的平均回报来计算优势值。

$$\frac{1}{k}\sum_{i=1}^{k}\left[R(y^{(i)},x) - \frac{1}{k-1}\sum_{j\neq i} R(y^{(j)},x)\right]\nabla \log \pi(y^{(i)}|x), \quad (3)$$

$y^{(i)}$ 是从 $\pi_\theta(\cdot \mid x)$ 生成的独立样本。为了确保训练的稳定性，我们为生成设置了一个最大标记限制，并包含提示：”请逐步推理，并将你的最终答案放在盒装的 中。”这种设置确保了如果输出过长，将不会生成最终答案，这被视为不正确。我们为正确答案分配奖励，而不正确的答案则获得零奖励。我们的奖励函数定义如下：

$$R(y^{(i)},x) = \begin{cases} f(\text{len}(y^{(i)})), & \text{if } y = y^*, \\ 0, & \text{otherwise.} \end{cases} \quad (4)$$

2.3 长度惩罚

由于仅凭二元正确性无法体现推理的效率，我们引入了基于长度的惩罚来鼓励简洁的解决方案。给定答案 $y^{(i)}$，将 $\text{len}(y^{(i)})$ 表示为 i 个预测序列的长度。我们定义的加权长度惩罚（PLP）公式如下：

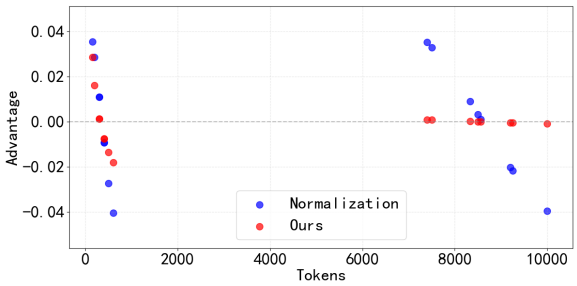
$$f(\text{len}(y^{(i)})) = 1 + \frac{\alpha}{\text{len}(y^{(i)})^\gamma}, \quad (5)$$

其中 $\alpha \geq 0, \gamma > 0$ 是超参数。在 PLP 下，较长的回复会受到相对较小的惩罚，而较短的回复会受到更严重的惩罚。这种设计与我们期望的行为一致：简单问题倾向于简洁，复杂问题则优先考虑准确性。我们选择 PLP 而不是标准化惩罚方法，因为后者倾向于使不同问题难度的响应长度变得标准化，从而降低对生成答案实际长度的敏感性。结果是，很难控制推理效率与正确性之间的权衡。相反，PLP 通过直接将响应长度融入惩罚项，允许更细粒度的控制，使模型能够根据问题的复杂性调整其推理深度。

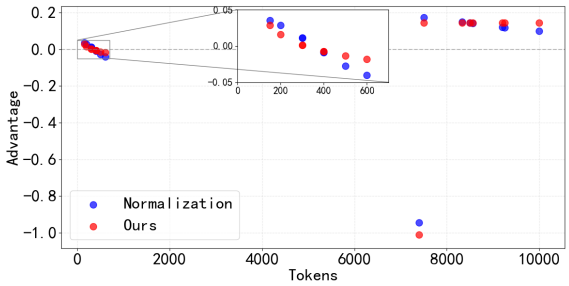
在 RLOO 算法中，优势函数与奖励信号紧密相连。通过收集多个在线样本，该方法实现了无偏差的方差缩减。在这一框架下，每个样本都可以作为其他样本的基准。通过从每个样本的奖励中减去这个基准，我们计算出其对应的优势值。实质上，这一过程评估了给定样本相对于批次中其他样本的优劣程度，从而实现更加稳定和高效的策略更新。

在我们的方法中，我们采用了一个加权长度惩罚（PLP），对较长的回答给予较温和的惩罚，以便对困难问题进行必要的推理。相比之下，像 Arora and Zanette (2025) 所提出的方法使用标准化的长度惩罚，这使得惩罚在样本之间标准化。这种标准化导致较长回答的奖励值几乎不可区分——尤其是在长度分布相似的情况下——从而降低了模型区分长而有意义的输出的能力。此类方法往往过度抑制长回答，即使这些回答对于解决复杂问题是必要的，这与我们在需要时鼓励更深入推理的目标相冲突。

为了刻画基于长度的惩罚的统计行为，我们考虑一种简化的情形，其中序列长度 $\text{len}(y)$ 在区间 (a,b) 上均匀分布，即 $\text{len}(y) \sim \mathcal{U}(a,b)$。在这项研究中，我们将讨论限制为 γ 等于 0.5 的情况。为了方便分析，设置缩放系数为 1 的



(a) 当所有回答都是正确时的优势。



(b) 当一个答案是错误时的优势。

Figure 3: 跨两个示例范围：300-600 和 7,000-10,000 个 tokens，标准化长度惩罚方法和绝对长度惩罚方法的比较。蓝色表示标准化方法，而红色表示绝对方法。

情况下，

$$Z = 1 + \frac{1}{\sqrt{\text{len}(y)}},$$

。在此假设下，Z 的方差为：

$$\text{Var}(Z) = \frac{\ln b - \ln a}{b - a} - \frac{4}{(\sqrt{a} + \sqrt{b})^2}. \quad (6)$$

由于对 a 和 b 的偏导数均为负，当 a 和 b 增加时，方差相应减小。也就是说，实现了在简单问题中具有较大长度惩罚差异，而在困难问题中几乎没有长度惩罚差异的目标。然而，标准化样本的方差始终保持为 1。即使有惩罚系数，方差仍保持为固定值，并不随长度变化而变化。这两种不同方法的方差在一般分布中也表现出相同的趋势。从图 3 中可以直观地观察到，当答案长度在 300 到 600 之间并且答案全部正确时，两种方法对不同长度答案的奖励值存在显著差异。然而，当答案长度在 7000 到 10000 之间时，我们的方法对正确答案几乎没有长度惩罚，而高效方法仍有相对较大的长度惩罚。尽管由于我们模型在给出长答案时奖励值差异较小而导致不同答案间的优势差异相对较小，但这是在所有答案都正确的情况下。当回答的问题中出现错误，即出现奖励为 0 时，优势值的差异将显现。这与我们处理困难问题时的目标一致，即更强调答案的正确性，而不是答案的长度。

3 实验

在本节中，我们提供实验结果以评估我们方法的有效性。模型我们在 DeepSeek-R1 和 Qwen2.5 上进行了实验。在我们的实验中，我们使用了这些家族中的三个模型：DeepSeek-R1-Distill-Qwen-1.5B、DeepSeek-R1-Distill-Qwen-7B 和 Qwen2.5-7B-Instruct。其中，前两个是推理模型，Qwen2.5-7B-Instruct 是非推理模型。我们希望了解我们的方法是否对推理和非推理模型都有效。数据集用于训练的数据集是 GSM8K，它是一个 8.5K 高质量语言多样化的小学数学词问题数据集。创建该数据集的目的是支持需要多步骤推理的基础数学问题的问答任务。为了训练，我们从 GSM8K 训练集中选择了 3200 个问题。对于每个模型，我们为每个问题生成了 8 个解决方案。测试数据集包括 GSM8K、MATH500 和 AIME2024，涵盖了不同难度级别的各种数学问题。基准在我们的实验中，我们采取了以下方法作为基准。在这种方法中，我们在评估原始模型时加入提示，比如“请尽可能少输出。”以缩短思维链的长度。(2) Training Language Models to Reason Efficiently 该方法对每个样本的多个答案进行标准化，使用 sigmoid 函数将结果值映射到 $[0, 1]$ 的范围，并通过系数 α 调节惩罚力度。

实现细节对于 1.5B 模型，我们使用 2 个 A100 GPU 进行训练，对于 7B 模型，我们使用 8 个 A100 GPU 进行训练。训练期间我们使用 vllm 限制模型输出的最大长度。由于我们用于训练的数据集是相对简单的 GSM8K，我们将生成长度限制设置为 2000 个标记。训练精度设置为 bfloat16。所有模型使用的批量大小为 128，并且每次迭代我们从数据集中选择 32 个提示。每个提示生成 8 个响应。我们为所有模型设置的学习率是 5×10^{-6} 。对于每个参数设置，我们进行三次独立的训练并分别评估每个模型，报告平均结果。每次实验运行大约需要两个小时完成。

评估配置我们遵循之前的工作，将所有模型的最大生成长度定义为 37268（包括思考标记和答案标记）。对于像 Deepseek 的模型，在评估过程中我们使用模型的官方模板。对于每个测试问题，我们以 0.6 的温度和 0.95 的最高概率值进行条件采样以获得 N 个输出，然后报告这些 N 个输出的平均准确率。具体来说，对于包含 1319 个测试样本的 GSM8K，我们设置 N 为 1，对于有 500 个样本的 MATH500，我们设

Models		GSM8K		MATH500		AIME2024	
DeepSeek-R1		Accuracy	Tokens	Accuracy	Tokens	Accuracy	Tokens
1.5B	Original	76.5 %	720	82.9 %	5446	27.7 %	15643
	PE	75.1 %	738	83.1 %	5054	33.3 %	15892
	Efficient	85.4 %	702	83.2 %	3641	33.3 %	15087
	Ours	86.3 %	411	85.1 %	3606	33.7 %	13327
7B	Original	92.6 %	1629	92.4 %	4141	54.7 %	12816
	PE	87.7 %	619	91.4 %	3614	49.7 %	13783
	Efficient	89.2 %	226	90.7 %	2329	48.7 %	9721
	Ours	90.1 %	218	91.3 %	1906	55.7 %	9056

Table 1: 模型性能比较，PE 代表提示工程。Efficient 是 (Arora and Zanette, 2025) 中使用的方法。

置 N 为 3，对于只有 30 个样本的 AIME2024，我们设置 N 为 10。

3.1 结果

如表格 1 所示，在 1.5B 模型中，我们模型在每个数据集上的准确性相比于原始模型都有所提升。此外，输出标记的数量也在一定程度上有所减少。对于 7B 模型，尽管在简单数据集上的准确性有所下降，但标记数量有显著减少。例如，在 GSM8K 数据集中，标记数量从 1629 减少到了 218，接近 88 % 的减少。而对于最难的数据集 AIME2024，准确率略有提高，同时标记数量减少。提示工程往往不稳定，在一些测试中，甚至会生成比不使用它时更多的标记，这与强制截断的设定范围相关联。虽然高效的方法同样可以在减少标记数量的同时提高准确性，但其效果不如我们的方法。而且它的稳定性较差，如图 5 和图 6 所示。对于 1.5B 模型，当 a 值设为 0.4 时，可以观察到 MATH500 和 AIME2024 数据集中的标记数量急剧下降，同时伴随着准确率的显著下降。这种情况也出现在 7B 模型中。相比之下，对于我们的模型，当标记数量达到相同数量级时，准确率的下降更加缓慢。

在表 2 中，我们还在 GSM8K 上进行了 Qwen 的实验比较。对于类似 Qwen 的模型，我们使用了 Qwen2.5-7B-Instruct 模型进行实验。我们比较了原始模型、我们的方法以及由 Tokenskip 方法训练的模型。可以看出，对于我们的模型，虽然准确率仅下降了 1 %，但令牌数减少了 40 % 。与 TokenSkip 相比，我们的方法准确率的下降更低，但令牌数的减少更大。

3.2 实证分析

模型参数规模的差异我们观察到，在 1.5B 和 7B 模型中，带有长度惩罚的强化学习减少了模型输出的令牌数。然而，减少的程度并不

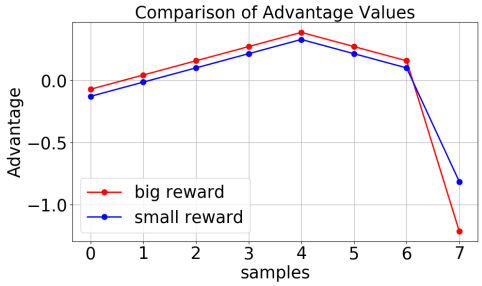


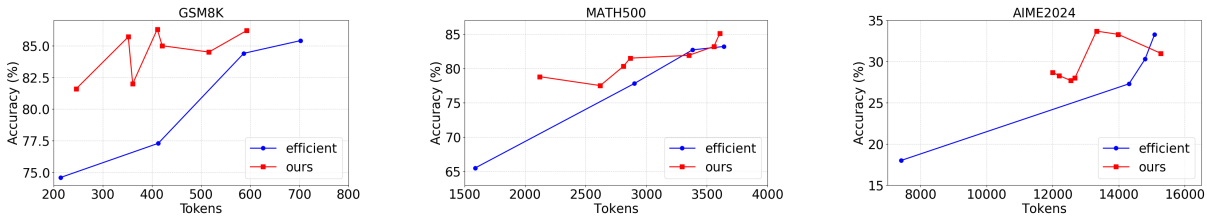
Figure 4: 当最后一个样本不正确时，大奖励和小奖励之间的区别。

相同。显然，在 1.5B 模型中，由强化学习带来的长度惩罚要比在 7B 模型中小得多。我们认为这是因为 7B 模型足够强大，并且训练数据集 GSM8K 中的问题对于 7B 模型来说过于简单。因此，即使推理长度减少了 90 % ，仍然能够得到正确的答案。这一趋势也导致了 MATH500 和 AIME2024 数据集的推理链显著减少。这就是为什么 7B 模型在准确性方面的优化不如 1.5B 模型显著的原因。

长度惩罚策略的差异虽然大多数现有策略通过从常数如 (Aggarwal and Welleck, 2025; Team et al., 2025; Xiao et al., 2025) 中减去惩罚来施加惩罚，但我们的方法通过将惩罚加到常数 1 来引入惩罚。由于优势函数仅依赖于组内奖励值之间的相对差异，而不是其绝对大小，这种修改实现了类似的长度惩罚效果。为了清晰起见，我们将两种策略分别称为“大奖励”和“小奖励”。如果我们将两种惩罚限制在区间 [0.6, 1] 和 [1, 1.4]，当基础分布一致时，由 RLOO 算法产生的优势值保持一致。然而，在出现错误时——例如在图 4 中的第八个样本，奖励降为零——我们的方法相较于高效方法，在正确和错误的输出之间产生了明显更大的分离。这种更大的分离解释了即使在 token 预算显著减少

Models	Accuracy	Average Tokens	Compression rate
Original	91.4 %	298	1.00
Tokenskip	89.9 %	217	0.70
Ours	90.3 %	182	0.61

Table 2: Qwen2.5-7B-Instruct 在 GSM8K 上的模型效率结果。Tokenskip (Xia et al., 2025) 的结果指的是参考论文中压缩率为 0.7 的情况。



(a) 1.5B-GSM8K (b) 1.5B-MATH500 (c) 1.5B-AIME2024

Figure 5: 我们的方法和高效方法的区别。对于我们的方法，系数是 1, 2, 3, 4, 5, 20, 30，而对于高效方法，系数是 0.05, 0.1, 0.2, 0.4。

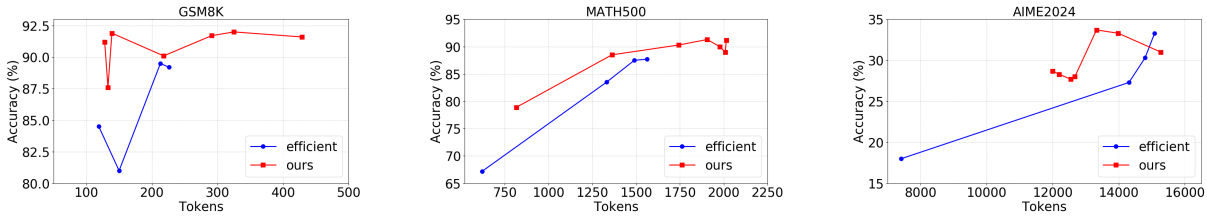
的情况下，我们的准确性仍能保持相对稳定的原因。相比之下，在小奖励方案中，过高的惩罚系数可能会将一个长且正确答案的奖励推向零。因此，一些正确输出会导致不成比例的大惩罚，降低其被选择的概率；一旦答案长度低于模型的容量阈值，准确性就会急剧下降。

思维消失在这个实验中，我们使用的所有 Deepseek-R1 类模型都是推理模型。它们会在产生输出之前先进行思考。然后，以 <\think><\think> 的形式，它们会在回答问题前将其用作思考的内容。我们惊讶地发现，GSM8K 中的大多数回答没有思考过程的标签。然而，对于困难的问题，模型仍然会有思考过程的标签。我们对原始模型和我们模型在三个不同难度数据集中的 CoT 数量进行了统计分析，如表格 **??** 所示。这意味着模型有能力判断问题是否足够简单，从而无需经过思考过程就能正确回答。这与我们的预期目标一致。我们比较了原始模型和训练模型中包含 CoT 响应的数量，并对 CoT 词元占总响应词元的比例进行了统计分析。

我们进行了一个消融实验。通过选择不同的参数，我们进行了多次实验。如图 5 和图 6 所示，在 1.5B 模型上，我们选择了 1, 2, 3, 4, 5, 20 和 30 作为实验参数。对于 7B 模型，我们选择了 1, 2, 3, 4, 5, 8 和 10 作为实验参数。我们发现，随着 α 值的增加，标记的数量总体上呈现下降趋势。在 1.5B 模型中，我们发现选择 $\alpha = 2$ 得到了最佳结果，而在 7B 模型中，最佳性能是在 $\alpha = 4$ 时实现的。我们尝试使用其他强化学习算法，例如 GRPO。然而，GRPO 算法在计算优势时会对奖励进行标准化。这导致当所有结果都是正确的时候，优势值变得很小。在除以标准差后，奖励被显著放大。因此，当加入惩罚项时，训练过程中答案长度迅速缩短。这使得模型完全关注答案长度而非准确性。因此，我们决定放弃这种方法。

4 结论

在本文中，我们提出了一种方法。通过在奖励函数中对答案长度添加绝对惩罚，我们可以使模型的推理更高效。也就是说，我们可以在简单问题上减少甚至消除思维链，同时更多关注困难问题答案的准确性。我们在 GSM8K 数据集上进行训练，并在三个不同难度级别的数据集上进行测试，分别是 GSM8K，MATH500 和 AIME2024。结果表明，我们的方法在缩短答案长度的同时，在简单和中等数据集上保持几乎相同的准确性。对于困难数据集，尽管答案长度的缩短幅度相对较小，但与原始模型相比，准确性有所提高。由于资源限制，我们使用的最大模型是一个 7B 模型，因此我们的方法尚未在更大参数的模型上验证。同时，由于计算资源的限制，模型训练期间设置的生成数量为 2000，这限制了我们使用更困难的训练数据集。未来，我们将使用更多的资源在上述设置下进行实验。此外，我们只在数学数据集上进行了测试。下一步，我们将尝试在其他领域的数据集上进行测试，以观察训练的模型是否具有泛化能力。



(a) 7B-GSM8K (b) 7B-MATH500 (c) 7B-AIME2024

Figure 6: 我们的方法与高效方法的区别。在我们的方法中，系数为 1, 2, 3, 4, 5, 8, 10，而在高效方法中，系数为 0.05, 0.1, 0.2, 0.4。

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Pranjal Aggarwal and Sean Welleck. 2025. L1: Controlling how long a reasoning model thinks with reinforcement learning. *arXiv preprint arXiv:2503.04697*.

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. 2024. Back to basics: Revisiting reinforce style optimization for learning from human feedback in llms. *arXiv preprint arXiv:2402.14740*.

Daman Arora and Andrea Zanette. 2025. Training language models to reason efficiently. *arXiv preprint arXiv:2502.04463*.

Mengru Ding, Hanmeng Liu, Zhizhang Fu, Jian Song, Wenbo Xie, and Yue Zhang. 2024. Break the chain: Large language models can be shortcut reasoners. *arXiv preprint arXiv:2406.06580*.

Xidong Feng, Ziyu Wan, Muning Wen, Stephen Marcus McAleer, Ying Wen, Weinan Zhang, and Jun Wang. 2023. Alphazero-like tree-search can guide large language model decoding and training. *arXiv preprint arXiv:2309.17179*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Tingxu Han, Zhenting Wang, Chunrong Fang, Shiyu Zhao, Shiqing Ma, and Zhenyu Chen. 2024. Token-budget-aware llm reasoning. *arXiv preprint arXiv:2412.18547*.

Daniel Kahneman. 2011. *Thinking, fast and slow*. macmillan.

Ayeong Lee, Ethan Che, and Tianyi Peng. 2025. How well do llms compress their own chain-of-thought? a token complexity approach. *arXiv preprint arXiv:2503.01141*.

Tengxiao Liu, Qipeng Guo, Xiangkun Hu, Cheng Jiayang, Yue Zhang, Xipeng Qiu, and Zheng Zhang. 2024. Can language models learn to skip steps? *arXiv preprint arXiv:2411.01855*.

Haotian Luo, Li Shen, Haiying He, Yibo Wang, Shiwei Liu, Wei Li, Naiqiang Tan, Xiaochun Cao, and Dacheng Tao. 2025. O1-pruner: Length-harmonizing fine-tuning for o1-like reasoning pruning. *arXiv preprint arXiv:2501.12570*.

Xinyin Ma, Guangnian Wan, Runpeng Yu, Gongfan Fang, and Xinchao Wang. 2025. Cot-valve: Length-compressible chain-of-thought tuning. *arXiv preprint arXiv:2502.09601*.

Tergel Munkhbat, Namgyu Ho, Seo Hyun Kim, Yongjin Yang, Yujin Kim, and Se-Young Yun. Self-training elicits concise reasoning in large language models, 2025. *URL https://arxiv. org/abs/2502.20122*.

Ethan Perez, Patrick Lewis, Wen-tau Yih, Kyunghyun Cho, and Douwe Kiela. 2020. Unsupervised question decomposition for question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8864–8880, Online. Association for Computational Linguistics.

Yuxiao Qu, Matthew YR Yang, Amrith Setlur, Lewis Tunstall, Edward Emanuel Beeching, Ruslan Salakhutdinov, and Aviral Kumar. 2025. Optimizing test-time compute via meta reinforcement fine-tuning. *arXiv preprint arXiv:2503.07572*.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew Wen, Shaochen Zhong, Hanjie Chen, et al. 2025. Stop overthinking: A survey on efficient reasoning for large language models. *arXiv preprint arXiv:2503.16419*.

Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, et al. 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.

Heming Xia, Yongqi Li, Chak Tou Leong, Wenjie Wang, and Wenjie Li. 2025. Tokenskip: Controllable chain-of-thought compression in llms. *arXiv preprint arXiv:2502.12067*.

Wenyi Xiao, Leilei Gan, Weilong Dai, Wanggui He, Ziwei Huang, Haoyuan Li, Fangxun Shu, Zhelun Yu, Peng Zhang, Hao Jiang, et al. 2025. Fast-slow thinking for large vision-language model reasoning. *arXiv preprint arXiv:2504.18458*.

Silei Xu, Wenhao Xie, Lingxiao Zhao, and Pengcheng He. 2025. Chain of draft: Thinking faster by writing less. *arXiv preprint arXiv:2502.18600*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822.

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*.

在我们的训练中，我们为 DeepseekR1 类模型和 Qwen 类模型使用了不同的输入模板，具体如下：
DeepseekR1 的输入模板：
<|begin_of_sentence|><|User|> 请逐步推理，并将最终答案放入 \框中{{}}。问题：{}<|Assistant|>

<|im_start|>system \n 你是一个有帮助的助手。<|im_end|> \n<|im_start|>user \n 请逐步推理，并将最终答案放入 \\框中{{}}。\n{}<|im_end|> \n<|im_start|>assistant \n

我们将以表格形式呈现所有超参数的结果，其中 Models 表示所使用的模型，Tokens 代表输出的标记数量，Acc 表示准确率。相同的参数在 GSM8K、MATH500 和 AIME2024 上进行了测试。

Model	α	Tokens	Acc
R1-1.5B	1	516	84.5 %
R1-1.5B	2	593	86.2 %
R1-1.5B	3	421	85.0 %
R1-1.5B	4	352	85.7 %
R1-1.5B	5	411	86.3 %
R1-1.5B	20	246	81.6 %
R1-1.5B	30	205	79.8 %

Table 3: 1.5B-GSM8K

Model	α	Tokens	Acc
R1-1.5B	1	3558	83.2 %
R1-1.5B	2	3606	85.1 %
R1-1.5B	3	2867	81.5 %
R1-1.5B	4	2811	80.3 %
R1-1.5B	5	3350	81.9 %
R1-1.5B	20	2620	77.5 %
R1-1.5B	30	2569	74.6 %

Table 4: 1.5B-MATH500

Model	α	Tokens	Acc
R1-1.5B	1	15275	31.0 %
R1-1.5B	2	13986	33.3 %
R1-1.5B	3	12000	28.7 %
R1-1.5B	4	12673	28.0 %
R1-1.5B	5	13327	33.7 %
R1-1.5B	20	12192	28.3 %
R1-1.5B	30	12391	21.7 %

Table 5: 1.5B-AIME2024

Model	α	Tokens	Acc
R1-7B	1	429	91.6 %
R1-7B	2	325	92.0 %
R1-7B	3	182	91.2 %
R1-7B	4	218	90.1 %
R1-7B	5	291	91.7 %
R1-7B	8	139	91.9 %
R1-7B	10	133	87.6 %

Table 6: 7B-GSM8K

Model	α	Tokens	Acc
R1-7B	1	2015	91.2 %
R1-7B	2	1977	90.0 %
R1-7B	3	1744	90.3 %
R1-7B	4	1906	91.3 %
R1-7B	5	2009	89.0 %
R1-7B	8	1362	88.5 %
R1-7B	10	816	78.9 %

Table 7: 7B-MATH500

Model	α	Tokens	Acc
R1-7B	1	9301	55.0 %
R1-7B	2	8383	53.7 %
R1-7B	3	10863	55.7 %
R1-7B	4	9058	55.7 %
R1-7B	5	8458	54.0 %
R1-7B	8	9618	50.7 %
R1-7B	10	7242	42.3 %

Table 8: 7B-AIME2024