

SABER: 可切换和平衡的训练以提高大型语言模型推理效率

Kai Zhao^{1*}, Yanjun Zhao^{1*}, Jiaming Song¹, Shien He¹, Lusheng Zhang¹,
Qiang Zhang¹, Tianjiao Li^{1†}

¹Bilibili Inc. { zhaokai02, zhaoyanjun01, songjiaming, heshien, zhanglusheng01, zhangqiang, litianjiao01 }
@bilibili.com

Abstract

由链式思维推理赋能的大型语言模型 (LLMs) 在复杂任务上取得了令人印象深刻的准确率, 但在应用于所有问题时, 其过高的推理成本和延迟却成为障碍。我们提出了 SABER (用于高效 LLM 推理的可切换和平衡训练), 这是一种赋予 LLMs 用户可控、基于令牌预算的推理的强化学习框架。SABER 首先分析每个训练样本的基础模型思维令牌使用情况, 并将其分配到预定义的预算层级之一。在微调过程中, 模型通过系统提示和长度感知奖励来遵循其分配的预算。同时, 我们结合无需思考的示例以确保即便关闭了明确的推理, 模型依然可以保持可靠。SABER 进一步支持四种离散的推理模式——NoThink, FastThink, CoreThink 和 DeepThink, 从而在延迟和推理深度之间实现灵活权衡。在数学推理 (MATH, GSM8K)、代码生成 (MBPP) 和逻辑推理 (LiveBench-Reasoning) 上的广泛评估表明, SABER 在紧缩预算下实现了高准确率、优雅退化, 以及有效的跨尺度和跨领域泛化。特别是, SABER-FastThink 将推理长度减少了 65.4%, 并在 MATH 基准上比基础模型提高了 3.6% 的准确率。

介绍

最近在大型语言模型 (LLMs) (Achiam et al. 2023; Bai et al. 2023) 的进展显著提高了它们处理复杂推理任务的能力, 能够通过明确的逐步思考来进行推理。像 Chain-of-Thought 提示 (Wei et al. 2023) 和测试时推理扩展 (Snell et al. 2024) 等方法允许模型在给出最终答案之前把问题分解为中间步骤。这些策略已经在多个领域中证明了其有效性。然而, 它们也带来了新的挑战: 推理过程往往变得过于冗长, 导致推理成本和延迟增加。更重要的是, 这种推理行为通常在所有问题中被统一应用, 而不考虑任务复杂性或用户偏好。

这种推理深度与任务要求之间的不匹配导致了日益受到关注的过度思考问题, 其中大型语言模型 (LLM) 甚至在处理微不足道的输入时也产生了不必要的复杂推理。例如, 有些模型可能在回答“1 + 1 等于几?”时探讨多种加法策略, 包含无关的理由, 并消耗比需要多出数十倍的符号。这不仅会导致响应速度变慢, 还会增加服务成本, 限制部署效率。尽管之前的工作 (Aggarwal and Welleck 2025; Li et al. 2025b; Kimi 2025) 曾试图通过指令调整、响应长度控制或奖励塑造来缩短输出, 这些方法通常施加了严格的约束或者依赖于与任务无

*These authors contributed equally.

†Corresponding author.

关的启发式方法。它们缺乏根据问题难度动态调整推理长度的能力, 或给用户显式控制推理过程的能力。

为了克服这些限制, 我们提出了 SABER——用于高效 LLM 推理的可切换和平衡训练方法, 这是一个强化学习框架, 能够使语言模型在明确指定的模式下进行推理。与应用统一约束不同, SABER 首先分析基础模型的输出, 以估计每个样本所需的推理工作量, 然后分配反映任务难度的分层标记预算。在 RL 训练过程中, 模型被引导遵循其目标预算, 使用系统提示和长度感知的奖励塑形。这种方法在保持复杂示例的长形式推理能力的同时, 鼓励在简单输入上进行高效推理。

特别是, SABER 定义了四种离散推理模式——NoThink、FastThink、CoreThink 和 DeepThink, 这允许在推理时显式控制推理的粒度。SABER 的一个关键特征是它对启用思考和禁用思考模式的统一处理。虽然之前的工作假设推理总是处于活动状态, 但实际应用通常需要在没有思考过程的情况下立即响应。我们明确加入了一组精心挑选的禁用思考的例子, 以确保在这种情况下的性能。因此, SABER 可以在单一模型中同时支持深思熟虑和直接响应风格, 减少思考和不思考模式之间的性能差距。此外, 这些模式不仅可解释且用户可控, 还能很好地跨域泛化。与之前只专注于数学推理任务的工作 (Li et al. 2025b; Huang et al. 2025a) 不同, SABER 同时在数学推理和代码生成任务中进行训练。我们进一步表明, SABER 可以扩展到更大的模型, 并且所学习的推理行为能够有效地迁移到未见过的逻辑推理任务中, 这突显了 SABER 的强大跨规模和跨域泛化能力。

此外, SABER 的训练过程非常高效且稳定。虽然许多之前基于强化学习的高效推理方法依赖监督微调 (SFT) 作为必要的热启动, 但 SABER 可以直接通过强化学习从经过蒸馏的基础模型进行优化。它的结构感知设计、课程令牌预算和多部分奖励公式允许其快速收敛而无需预训练开销。本研究的主要贡献总结如下:

- 我们提出了一个统一的框架 SABER, 该框架能够高效且稳定地优化长思维语言模型, 通过基于预算的数据分组、课程式退化和奖励约束实现高效性和训练稳定性的双重目标。
- SABER 支持四种推理模式——NoThink、FastThink、CoreThink 和 DeepThink, 允许用户明确控制模型的推理深度。这种统一且可切换的设计适应了多种使用场景, 从最小延迟响应到复杂推理。
- 在数学推理、代码生成和逻辑推理基准上的综合实

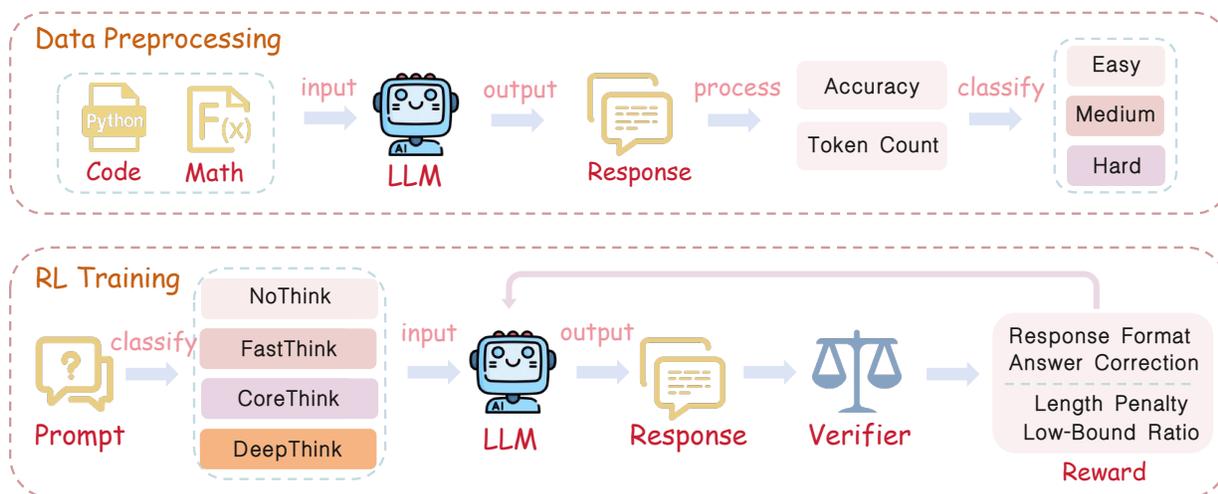


Figure 1: 对 SABER 框架的概述。上半部分展示了数据预处理管道，其中推理预算通过基础模型推断进行估计，并用于根据难度（简单/中等/困难）对训练样本进行分类。下半部分显示了强化学习训练过程，其中特定模式的提示指导模型生成响应，然后根据格式正确性、答案准确性和长度对齐性进行多部分奖励评估。

验表明，SABER 在严格的 token 预算下能够保持强大的性能，同时也在各种任务间实现优雅的性能下降和广泛的泛化能力。

相关工作

大规模推理模型中的过度思考问题。有几项研究 (Sui et al. 2025; Feng et al. 2025; Hou et al. 2025; Lu et al. 2025; Zhuang, Wang, and Sun 2025; Lee, Che, and Peng 2025) 探讨了在不牺牲强化学习微调所带来的逻辑收益的情况下压缩思维链的想法。早期关于 L2S 的研究 (Wu et al. 2025a) 表明，将“长思考者”策略与简洁的“仅答案”策略合并，可以产生一个在保持准确性的同时，发出更少的令牌的单一模型。在这个见解的基础上，混合蒸馏 (Chenglin et al. 2024) 通过联合蒸馏详细和简洁的推理，从强大的教师中传递知识，然后应用一个长度感知的 RL 目标，以便学生可以决定何时简洁即可。一条平行的方向通过直接将长度因素纳入偏好评分来重新设计直接偏好优化 (DPO) (Liu et al. 2024)，使模型能够在正确时偏爱短证明，并仅在必要时依赖更长的推理。这些思想在 Kimi Long2Short (Kimi 2025) 中融合，该方法结合了混合蒸馏、对长度敏感的 DPO 和拒绝抽样，在生产聊天中实现了多倍加速，同时在数学基准上与其长格式教师相匹配。最后，L1 (Aggarwal and Welleck 2025) 通过在训练期间根据显式的令牌预算调节策略来推广该范式；在推理时，同一个网络可以沿着成本-准确性曲线“滑动”，在延迟关键使用场景中遵循严格的限制或在需要全精度推理时放宽限制。

大型推理模型中的思维预算。其他研究将思维预算的概念直接嵌入到优化目标 (Shen et al. 2025; Wu et al. 2025b; Li et al. 2025a; Jiang et al. 2025; Fan et al. 2025) 中。SelfBudgeter (Li et al. 2025b) 添加了一个特殊标记，提示模型预测自身的预算；随后一种 RL 奖励在正确性与任何预算超支之间取得平衡，在 GSM8K 上将冗余推理减少一半以上。与自预测不同，Token-Budget-Aware LLM 推理 (TALE) (Han et al. 2025) 使用由监督者提供的预算进行训练，该预算近似于解决每个实

例所需的最小标记数量，加强了在答案明确后即停止的习惯。AdaCtrl (Huang et al. 2025a) 引入了一个两阶段方案，其中模型首先自我评估问题的难易程度，然后通过一个学习控制器将该难度映射到一个自适应预算，从而在没有外部提示的情况下显著加速。进一步推动自适应性，自适应长度惩罚 (ALP) (Xiang et al. 2025) 在线调节惩罚系数，只有在观察到的成功率证明额外计算是合理时才允许更多标记；这个简单的扩展在代码生成任务中减少了三分之一的标记量，同时略微提高了 pass-@-1 的准确率。

方法

在本节中，我们描述了 SABER 框架的三阶段方法，该框架支持不同的用户控制推理模式。我们首先介绍如何将基础模型的思维标记统计信息提取并分类为离散预算级别。接下来，我们详细说明我们的样本分组和稳定性控制机制。然后，我们介绍无思考样本的注入，以启用直接回答模式。最后，我们形式化我们的强化学习目标，包括奖励组件，这些组件共同强制格式、答案正确性、长度对齐和防攻击约束。图 1 总结了 SABER 框架。

思维采集与预算分类

思维预算的设计是 SABER 的核心。如果每个例子的目标预算都被设置为相同的值，那么需要很少思考令牌的问题将永远不会遭受长度惩罚，因此无法学会如何切换推理模式，而需要大量思考令牌的问题将被持续惩罚并很快在性能上崩溃。为了解决这个问题，SABER 对每个例子应用预算校准。首先，我们在整个训练集中运行基础模型，并记录在 `<think>` 和 `</think>` 之间生成的令牌数量。根据观察到的分布和经验判断，我们将例子分为三个难度等级——128 (简单)、4,096 (中等) 和 16,384 (困难)，并为每个例子分配相应的目标预算，如下所示：

- 如果一个例子的思维标记数量少于 128，我们仍然将其目标预算设置为 128。

DeepThink

System message: [original sys]

Question: [original prompt]

Answer: <think> ... </think> [response]

CoreThink / FastThink

System message: [original sys] Your reasoning process between <think> and </think> should be STRICTLY UNDER XXX tokens.

Question: [original prompt]

Answer: <think> ... </think> [response]

NoThink

System message: [original sys] Respond directly without internal reasoning.\n<think></think>\n\n [response here]

Question: [original prompt]

Answer: [response]

Figure 2: SABER 不同模式的模板

- 如果它介于 128 和 4,096 之间，我们将其目标预算设为 128。
- 如果它处于 4,096 和 16,384 之间，我们将其目标预算设定为 4,096。
- 如果超过 16,384，我们不设置上限，从而允许模型在这些具有挑战性的任务上具有充分的推理自由。

然后，我们为每个提示添加一个系统消息，如图 2 所示，其中 XXX 被替换为例子的目标预算，而 DeepThink 模式没有目标预算。

这种分级降级策略提供了两个关键优势：(1) 它确保更多的训练样本从一开始就受到长度惩罚，从而有效地教授模型在不同推理模式之间切换；(2) 它尊重样本之间在推理长度上的固有变化，使相邻模式之间的过渡更加顺畅，并在训练效率与稳定性之间取得平衡。

用惩罚来稳定模式转换

在微调的一开始就对所有训练样本统一应用长度惩罚可能会引入两个关键问题。首先，频繁和突然的推理模式切换可能会导致训练不稳定。其次，模型可能会过于偏向生成较短的推理轨迹，导致思考不足和潜在的性能下降。为减轻这些风险，我们引入了两种互补的策略，以实现更平稳和更稳定的模式转换。

我们评估基础模型正确回答每个训练样本的能力。在基础模型未能解决的示例中（约占语料库的 40%），我们采取了两种处理方式：只有其余的 60% 示例是基础模型可以正确回答的，才会进行降级程序。这种分区方案确保了在训练早期阶段更保守地应用长度限制，从而促进推理模式切换的稳定学习。

为了避免模型只为了最小化惩罚而过度缩短其推理的退化行为，我们在生成推理长度上引入了额外的下界约束。具体来说，我们要求生成的 think-token 数量 t_{gen} 保持在基准模型的数量 t_{base} 的某一比例内。形式上，我们强制如下约束：

$$0.2 \cdot t_{\text{base}} \leq t_{\text{gen}} \leq 1.2 \cdot t_{\text{base}}.$$

这一范围确保模型在尊重目标预算的同时保持足够的推理内容，有效地阻止通过过度简洁进行的奖励操控。

用户控制的无思考模式

在实际应用中，用户有时可能更喜欢接收直接答案而没有任何中间推理 (Xiang et al. 2025; Chen et al. 2024)。然而，直接在长思考模型中禁用推理组件而没有任何针对性调整，通常会导致性能严重下降。这突显了明确训练模型以处理这种无思考场景的必要性。

为此，我们在训练语料库中增加了一部分特别构造的不思考例子。我们发现，即使是少量的这类数据也能够显著提高模型在不思考模式下的兼容性，从而缓解在故意跳过推理时性能的下降。每个不思考例子都是通过手动在输入中添加一个最小推理块构建的，如图 2 所示。这种格式明确指示模型绕过 <think> 范围并立即生成响应。通过学习这些例子，模型获得了优雅地处理用户直接回答请求的能力，为不同的使用场景提供了更大的灵活性。

直接强化学习优化，无需 SFT 预热

与许多以前的方法 (Huang et al. 2025b,a; Li et al. 2025b; Ma et al. 2025) 需要在强化学习 (RL) 之前进行监督微调 (SFT) 作为热启动不同，我们的 SABER 框架引入了与模型行为和训练动态自然对齐的操作。因此，SABER 不需要单独的 SFT 预热阶段，可以从一开始直接使用强化学习进行训练。这大大简化了训练流程并减少了计算开销。

在这项工作中，我们采用广泛使用的群体相对策略优化 (GRPO) 算法 (Shao et al. 2024)，通过结构化的奖励信号微调模型。GRPO 目标定义为：

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

奖励设计是强化学习优化的核心。如前所述，SABER 框架结合了由四个不同成分组成的复合奖励信号：

$$r = r_{\text{format}} + r_{\text{answer}} + r_{\text{length}} + r_{\text{ratio}}.$$

- 格式奖励：

$$r_{\text{format}} = \begin{cases} 0, & \text{if format is correct,} \\ -1, & \text{if format is wrong.} \end{cases}$$

格式奖励旨在确保模型的输出在句法上是可解析的。具体而言，它强制采用 <think> ... </think> ... 的结构化格式，以清晰地将模型的内部推理内容与最终答案区分开来。

- 答案奖励：

$$r_{\text{answer}} = \begin{cases} 1, & \text{if answer is correct,} \\ 0, & \text{if answer is wrong.} \end{cases}$$

答案奖励评估模型最终回答的正确性。对于数学问题，从 $\boxed{\quad}$ 中提取预测答案并与真实标签进行比较。对于代码生成任务，奖励通过提取用三重反引号 (“”) 包围的代码块，在安全的沙箱环境中执行，并根据预定义的测试用例进行验证来计算。

- 长度惩罚：

$$r_{\text{length}} = \begin{cases} 0, & \text{if } t_{\text{gen}} \leq t_{\text{budget}}, \\ -0.4, & \text{otherwise.} \end{cases}$$

其中 t_{gen} 是当前策略生成的思考标记的数量。该长度惩罚被设置为 -0.4，以将其与格式奖励区分开来。

- 下限比率惩罚：

$$r_{\text{ratio}} = \begin{cases} 0, & \text{if } 0.2 \cdot t_{\text{base}} \leq t_{\text{gen}} \leq 1.2 \cdot t_{\text{base}}, \\ -0.4, & \text{otherwise.} \end{cases}$$

，其中 t_{base} 表示由基本策略生成的思考标记数量。该约束防止模型滥用短或长的生成以进行奖励攻击。

通过联合优化这些奖励，我们的方法实现了与用户选择的推理模式精确对齐，在不同模式之间实现了平滑过渡，并在长推理、短推理和无推理场景中保持高质量的答案。

实验

在本节中，我们展示了 SABER 框架的实证评估，以回答以下研究问题 (RQs)：

RQ1: SABER 在数学推理和代码生成任务上如何与现有的强基线进行比较？

RQ2: SABER 能否推广到更大的模型和未见过的推理领域？

RQ3: 在 SABER 中的独立设计选择有多重要？

RQ4: 在 SABER 的可切换推理模式中出现了哪些定性差异？

为了解决这些问题，我们首先概述实验设置，包括基准、基线和训练数据。然后，我们在两个核心任务上使用 1.5B 模型展示主要结果：数学推理 (MATH/GSM8K) 和代码生成 (MBPP)。接下来，我们通过将相同的训练方法应用于 7B 模型和逻辑推理基准 (LiveBench-Reasoning)，展示跨规模和跨领域泛化能力。此外，我们通过逐一移除 SABER 的每个组件，进行消融研究。最后，我们对 FastThink、CoreThink 和 DeepThink 模式进行行为分析，突出了每种模式如何影响推理深度和答案准确性。

实验装置

基准测试。我们对数学推理和代码生成任务进行评估，包括以下内容：(1) MATH500 (Lightman et al. 2023) 是完整 MATH 基准测试中的一个 500 题的切片，旨在涵盖七个竞赛领域，同时保持紧凑以便于快速评估。(2) GSM8K (Cobbe et al. 2021) 由 8.5K 个众包的小学文字题组成，每个问题都可以通过几个算术步骤解决。(3) MATH (Hendrycks et al. 2021) 扩展到 12.5K 个 AMC/AIME 风格的竞赛问题，提供完整的逐步解决方案，涵盖代数、几何、组合数学等。(4) MBPP (Austin et al. 2021) 包含 974 个编程挑战，旨在测试 LLMs 生成解决算法问题代码的能力。(5) LiveBench-Reasoning 是 LiveBench (White et al. 2024) 的一个子集，用于评估模型的逻辑推理能力，包含 200 个复杂的逻辑谜题。

基线。我们在主要实验中使用 DeepSeek-R1-Distill-Qwen-1.5B 作为我们的基础模型。该模型是从 DeepSeek-R1 专门蒸馏而来的，并在同等规模的 LLM 中达到了新的最先进结果。我们选择 1.5B 的模型规模进行比较，因为该领域的许多相关工作采用了相同的规模，使其成为公平评估的标准参考点。为了评估 SABER 的有效性，我们还将其与两种相关方法进行比较：(1) L1 (Aggarwal and Welleck 2025)，提出了长度控制的策略优化，以生成符合提示中规定的严格长度限制的输出。(2) SelfBudgeter (Li et al. 2025b)，能够自主预测推理所需的 token 预算，并有效地遵循自我施加的约束。我们在更大的 7B 模型上进一步验证了 SABER。然而，由于这两种基线方法在此规模上没有发布结果，我们仅与基础模型 DeepSeek-R1-Distill-Qwen-7B 进行比较。对于我们实验中的 L1 和 SelfBudgeter 的评估结果，我们直接使用 SelfBudgeter 论文中报告的原始数据。对于 DeepSeek-R1-Distill-Qwen 和 SABER，我们使用 DeepSeek 推荐的推断配置进行评估，包括温度、top-p 和其他解码参数，以确保方法之间的公平和一致的比较。我们的训练实现基于开源的 verl 框架 (Sheng et al. 2024)，该框架可在：<https://github.com/volcengine/verl> 获取。

训练数据。SABER 的训练仅使用 2K 个样例：1K 数学实例和 1K 代码实例。具体而言，对于数学领域，我们首先使用数学验证器从 OpenR1-Math-220k 数据集中 (<https://huggingface.co/datasets/open-r1/OpenR1-Math-220k>) 过滤样本，只保留那些被验证为正确的，然后从验证后的子集中随机抽取 1K 实例作为 SABER 的训练数据。对于代码，我们从 KodCode-Light-RL-10K (<https://huggingface.co/datasets/KodCode/KodCode-Light-RL-10K>) 数据集中随机抽取 1K 个样例。我们将 SABER 在这 2K 个样例的组合集上训练 10 个周期，所得模型用于本文呈现的所有实验。这些数据通过基于准确性的策略进行划分，以促进推理模式切换的稳定学习。如图 1 所示，在数据预处理中，每个实例首先被标注上一个推理预算层级。为了结合 NoThink 模式，我们进一步为每个样本创建了一个重复项，并标注为零推理预算，与核心语料库完全重合。这确保了模型在相同的训练计划中遇到受限制和直接回答的变体。

主要结果

为了解答 RQ1，表格 1 对比了基线与我们 SABER 框架的变体，该框架结合了不同的思维模式。为了捕捉有效性和计算成本，我们报告了准确率 (Acc.) 和模型生成的平均思维长度 (Len.)。L1 通过在训练期间严格限制思维标记的数量，有效缩短思维链。然而，这一严格约束导致准确率显著下降。SelfBudgeter 通过预估合适的预算减轻了这种权衡，恢复了大部分失去的性能。然而，这需要对 3 个周期的 30K 示例进行多次训练，从而导致大量计算成本。这种低效源于缺乏任务感知的预算分配：由于所有样本被平等对待，而没有标记级别的退化，只有少数样本在训练期间受到有效的长度惩罚。因此，该方法需要大量的训练实例以实现收敛，显著增加了整体训练负担。

相比之下，SABER 仅在 2000 个样本上进行了训练，赋予模型评估问题难度的能力，展示了在学习切换推理模式方面的高效性。与基础模型 (DeepSeek-R1-Distill-Qwen-1.5B) 相比，SABER-FastThink 在平均推理长

Table 1: 主要性能比较针对 1.5B 大小的模型在数学和代码基准上的表现, 显示准确率 (Acc) 和平均响应长度 (Len)。

Model	MATH500		GSM8K		MATH		MBPP	
	Acc ↑	Len ↓	Acc ↑	Len ↓	Acc ↑	Len ↓	Acc ↑	Len ↓
Deepseek-R1-Distill-Qwen-1.5B	78.9	8042	82.1	3471	80.6	7866	38.9	5755
L1-Max (1.5B, Num Tokens=512)	-	-	72.9	569	71.2	1533	-	-
L1-Max (1.5B, Num Tokens=3600)	-	-	74.4	633	77.1	1545	-	-
SelfBudgeter (1.5B, slk, Train_Data=90k)	-	-	81.5	662	74.2	919	-	-
SABER - 1.5B - DeepThink	83.2	5353	85.7	1947	85.2	4748	53.7	3010
SABER - 1.5B - CoreThink	82.1	3294	83.1	930	84.3	3045	49.8	1254
SABER - 1.5B - FastThink	81.4	2899	82.5	778	83.5	2719	45.1	942
Deepseek-R1-Distill-Qwen-1.5B - NoThink	65.1	0	61.1	0	65.5	0	30.7	0
SABER - 1.5B - NoThink	76.3	0	78.1	0	76.9	0	44.7	0

Table 2: 在数学、代码和逻辑推理任务中, 对 SABER 进行跨尺度和跨领域泛化的评估, 显示准确率 (Acc)、平均响应长度 (Len)。

Model	MATH500		GSM8K		MATH		MBPP		LiveBench-R	
	Acc ↑	Len ↓	Acc ↑	Len ↓	Acc ↑	Len ↓	Acc ↑	Len ↓	Acc ↑	Len ↓
Deepseek-R1-Distill-Qwen-7B	91.5	8221	91.4	3402	92.4	7838	59.9	5021	36.4	5619
SABER - 7B - DeepThink	91.9	6265	92.0	2351	92.9	5763	65.8	2426	38.3	4401
SABER - 7B - CoreThink	90.1	2820	91.7	1319	91.2	2557	63.0	1556	32.1	3039
SABER - 7B - FastThink	86.5	1563	90.3	495	87.2	1478	61.1	969	30.6	2166
Deepseek-R1-Distill-Qwen-7B - NoThink	79.1	0	86.4	0	79.7	0	43.2	0	17.9	0
SABER - 7B - NoThink	85.1	0	89.2	0	85.5	0	58.8	0	26.8	0

度上减少了 72.7 %。值得注意的是, 这种简洁的推理并未降低准确性, 反而在所有基准测试中提高了平均 3.0 %, 展现了计算成本与性能之间更为有利的平衡。SABER-CoreThink 进一步改善了这种平衡, 将推理长度减少了 67.9 %, 同时将准确率提高了 4.7 %。SABER-DeepThink 则提供了最强的结果, 削减了 41.2 % 的推理长度, 并提高了 6.8 % 的准确率。这些进步可以归因于两个主要因素。首先, 更简洁的推理有助于避免不必要的重复或干扰, 这些可能会影响最终答案。其次, 我们观察到 SFT 训练的长推理模型倾向于表现出重复生成。在经过格式约束的强化学习后, 这种重复现象显著减少, 从而提升了基准测试表现。

跨尺度和跨领域泛化

为了回答 RQ2, 我们进一步将 SABER 应用于更大的 DeepSeek-R1-Distill-Qwen-7B 模型, 以评估其跨规模的泛化能力。结果如表 2 所示。在这种设置下, SABER-FastThink 保持了其效率优势, 将平均推理长度大幅减少了 82.1 %。然而, 这伴随着 2.5 % 的准确率下降, 反映了规模扩大会更明显的权衡。同时, SABER-DeepThink 在推理长度上减少了 33.2 %, 而准确率仅增加了 1.9 %。这些结果表明, 虽然性能因模型容量而异, 但我们的方法在不同模型规模上仍然有效且适应性强。此外, 虽然训练数据仅包括数学和代码示例, 但 SABER 的推理模式切换能力能够很好地泛化到未见的任务类型。表 2 显示了 SABER 在一个逻辑推理基准上的表现, 进一步证明了其强大的泛化能力。

消融实验

为了解答 RQ3, 我们进行了消融研究以评估 SABER 中关键组件的影响, 每个变体都仅修改 SABER 配置中的一个组件: (1) 全预算降级: 所有训练样本的推理级别都降级一级; (2) 无预算降级: 所有样本都严格在其基础模型的原始推理预算下进行训练; (3) 减少 NoThink 比例: 将训练中 NoThink 样本的比例减少到 30 %; (4) 移除 NoThink 数据: 从训练中移除所有 NoThink 模式的数据; 以及 (5) 无准确性过滤: 在预处理期间, 分配难度等级时不使用模型回答的正确性来过滤样本。

如表 3 所示, 移除预算降低会减缓对不同推理模式的适应, 而对所有样本采取激进的降低则会导致训练不稳定和准确性下降。此处, “无预算降低” 设置与基线 (Li et al. 2025b) 非常相似。没有渐进的预算引导, 模型在泛化较短推理模式时表现困难, 导致模式适应缓慢和效率降低。减少或去除 NoThink 样本显著损害了 NoThink 性能, 而未能改善其他模式, 确认了联合训练的必要性。最后, 移除基于准确性的过滤会导致监督噪声并降低稳定性。这些结果强调了 SABER 的所有单独组件对于有效和稳定学习都是必不可少的。

不同推理模式的行为

为了解答 RQ4, 我们从 MATH500 数据集中提供一个具有代表性的例子, 以说明三种思维模式之间的差异。如图 3 所示, 所有模式都遵循核心问题解决步骤: 分解 196 并计算其除数。这些步骤导致了正确的最终答案, 答案在图中用绿色突出显示。具体来说, 快思只包括这些基本步骤, 使推理过程简洁明了。相反, 核心思维在

Table 3: SABER 的各个组件在数学推理任务上的消融结果，显示了准确率 (Acc) 和平均响应长度 (Len)。

Training Setting	Test Setting	MATH500		GSM8K		MATH	
		Acc ↑	Len ↓	Acc ↑	Len ↓	Acc ↑	Len ↓
All Budget Downgrade	DeepThink	81.7	4902	84.8	1771	84.2	4579
	CoreThink	80.2	3663	82.8	850	82.8	3147
	FastThink	80.3	3042	81.0	674	82.0	2731
	NoThink	72.5	0	74.2	0	75.1	0
Without Budget Downgrade	DeepThink	83.3	5636	85.5	2182	84.9	5261
	CoreThink	81.7	5378	84.1	1763	84.4	4856
	FastThink	81.7	4896	83.3	1627	84.4	4583
	NoThink	75.1	0	74.3	0	75.7	0
Reduce NoThink Ratio	DeepThink	81.7	5072	85.2	1944	84.7	4957
	CoreThink	80.7	3816	82.7	977	83.1	3325
	FastThink	79.9	3424	81.8	847	82.5	3107
	NoThink	72.5	0	73.7	0	74.7	0
Remove NoThink Data	DeepThink	82.1	5317	84.9	1838	84.9	4767
	CoreThink	82.5	3819	82.7	1133	83.3	3545
	FastThink	81.0	3488	81.9	915	82.6	3154
	NoThink	61.4	0	58.2	0	62.3	0
Without Accuracy Filtering	DeepThink	83.7	5774	84.6	2210	85.0	5501
	CoreThink	83.0	4845	82.1	1446	84.2	4361
	FastThink	81.3	4254	80.5	960	83.7	4020
	NoThink	71.4	0	70.2	0	71.4	0

解决问题之前引入了一些初步推理，例如“我记得……”。在分解过程中，它还包括额外的中间步骤和偶尔的反思。此外，深思采用更全面和反思的推理过程。在得出解决方案后，它包括诸如“我认为这是正确的，因为……”之类的理由，随后进行最终的复查和自我验证。这表明了一种更深入和更全面的推理模式。

为了进一步研究 SABER 的可切换推理模式之间出现的定性差异，我们调查了反思标记的分布，计算与验证（“检查”、“验证”）、回顾（“回忆”）、分支探索（“或者”）、逻辑转折（“然而”、“因为”）和逐步分解（“逐步”）相关的提示词的标记级审查。正如图 4a 所示，FastThink 生成的反思性术语明显少于 DeepThink，在 GSM8K、MATH、MATH500 上分别减少了 80.24%、70.83% 和 61.97%。FastThink 在推理深度上达到了最显著的下降，但其答案准确性几乎保持不变。因此，FastThink 的策略是修剪填充词，同时保留指示真实推理步骤的过渡标记。

此外，我们还研究了不同推理模式的行为。图 4b 展示了三种推理模式的差异。与 DeepThink 相比，CoreThink 和 FastThink 缩短了思维链的长度。在基于 SABER (Deepseek-R1-Distill-Qwen-7B) 的 GSM8K (Cobbe et al. 2021) 上进行评估时，与 DeepThink 相比，FastThink 的推理标记平均数量下降了 79%。

结论与未来工作

在这项工作中，我们提出了 SABER，这是一种统一且可切换的推理框架，使大型语言模型能够在特定模式下执行高效且可控的推理。通过结合结构化奖励设计、离

散推理模式和课程式预算分配，SABER 在不需要监督的暖启动的情况下实现了高训练稳定性和推理灵活性。我们的实验表明，SABER 在数学推理、代码生成和逻辑推理任务中表现良好，即使在不同的计算约束下也能保持强大的性能。我们进一步展示了 SABER 在单一模型中有效支持思考模式和非思考模式，且性能降级最小。这些结果指出了在大型语言模型中实现可控和成本高效推理的一个有前途的方向。

尽管成绩令人鼓舞，但仍有一些注意事项。将框架扩展到异构的、实际的工作负载可能会暴露出我们的控制实验未涵盖的意想不到的实际障碍。该流程还依赖于一个评估 LLM 来计算每个提示的推理预算，并在随后的手动分组到难度带中。诸如替代解码策略等外部变量也可能影响方法的可迁移性。这些考虑因素不会削弱贡献，反而为未来的研究指明了明确的方向。

References

- Achiam, J.; Adler, S.; Agarwal, S.; Ahmad, L.; Akkaya, I.; Aleman, F. L.; Almeida, D.; Altenschmidt, J.; Altman, S.; Anadkat, S.; et al. 2023. Gpt-4 technical report. arXiv preprint arXiv:2303.08774.
- Aggarwal, P.; and Welleck, S. 2025. L1: Controlling How Long A Reasoning Model Thinks With Reinforcement Learning. arXiv:2503.04697.
- Austin, J.; Odena, A.; Nye, M.; Bosma, M.; Michalewski, H.; Dohan, D.; Jiang, E.; Cai, C.; Terry, M.; Le, Q.; and Sutton, C. 2021. Program Synthesis with Large Language Models. arXiv:2108.07732.
- Bai, J.; Bai, S.; Chu, Y.; Cui, Z.; Dang, K.; Deng, X.;

Question: How many positive whole-number divisors does 196 have?

FastThink

Reasoning Process: Okay, I need to find the number of positive whole-number divisors of 196. **First, I'll factorize 196** into its prime factors. Let's start by dividing 196 by 2: $196 \div 2 = 98$. Then, $98 \div 2 = 49$. Now, 49 is 7 squared, so $49 = 7^2$. **Putting it all together, $196 = 2^2 \times 7^2$.**

...
To find the number of divisors, I use the formula where I add one to each of the exponents in the prime factorization and multiply them. **So, $(2 + 1) \times (2 + 1) = 3 \times 3 = 9$.** Therefore, 196 has 9 positive divisors.

Final Answer: 9 (True)

CoreThink

Reasoning Process: Okay, so I need to figure out how many positive whole-number divisors 196 has. **Hmm, I remember that ... I should start by finding the prime factorization of the number.** Let me try that.
First, I know that 196 is an even number, so it's divisible by 2. Dividing 196 by 2 gives me 98. That's still even, so I can divide by 2 again to get 49. **Now, 49 is a number I recognize—it's 7 squared, right? So, 49 equals 7 times 7.** Putting that all together, the prime factors of 196 are 2 squared times 7 squared. **So, $196 = 2^2 \times 7^2$.**

...
Now, the formula for finding the number of divisors is to take the exponents of the prime factors, and then multiply those together. For 7^2 , the exponent is also 2, so adding one gives 3 as well.
Multiplying these together: 3×3 equals 9. So, 196 has 9 positive whole-number divisors.

Final Answer: 9 (True)

DeepThink

Reasoning Process: Okay, so I need to figure out how many positive whole-number divisors 196 has. **Hmm, ... I remember that ... Let me try to recall ... First, let me start by factoring 196...** I know that... Let's divide it by 2: $196 \div 2 = 98$. Okay... so... $98 \div 2 = 49$. Now... so $49 = 7 \times 7$. Therefore, putting it all together, **the prime factors of 196 are $2 \times 2 \times 7 \times 7$...** **I think that's correct because ...**

...
Now, the formula for finding the number of divisors is to ... So in this case ..., and then we can, so that...
Wait, let me make sure ... **So that's $(2+1)(2+1) = 3 \times 3 = 9$.** Yeah, that seems right.
Just to double-check, maybe I can list all the divisors of 196. Starting from ... Then ... And finally ... Let me count them: 1, 2, 4, 7, 14, 28, 49, 98, 196.
Hmm, so... That must be the correct answer... So yeah, 196 has 9 positive whole-number divisors.

Final Answer: 9 (True)

Figure 3: 对于相同的问题，SABER 支持四种思维模式。除了无思维模式外，其他三种模式包括核心解决步骤，例如分解和计数（用绿色显示）。CoreThink 独有的附加推理和反思用蓝色标记。DeepThink 中更深入的自我检查和论证步骤以棕色突出显示。

Fan, Y.; Ge, W.; Han, Y.; Huang, F.; et al. 2023. Qwen technical report. arXiv preprint arXiv:2309.16609.

Chen, X.; Xu, J.; Liang, T.; He, Z.; Pang, J.; Yu, D.; Song, L.; Liu, Q.; Zhou, M.; Zhang, Z.; Wang, R.; Tu, Z.; Mi, H.; and Yu, D. 2024. Do NOT Think That Much for $2+3=?$ On the Overthinking of o1-Like LLMs. arXiv:2412.21187.

Chenglin, L.; Chen, Q.; Li, L.; Wang, C.; Tao, F.; Li, Y.; Chen, Z.; and Zhang, Y. 2024. Mixed Distillation Helps Smaller Language Models Reason Better. In *AL-ONAIZAN, Y.; BANSAL, M.; and CHEN, Y.-N., eds., Findings of the Association for Computational Linguistics: EMNLP 2024*. Miami, Florida, USA: Association for Computational Linguistics.

Cobbe, K.; Kosaraju, V.; Bavarian, M.; Chen, M.; Jun, H.; Kaiser, L.; Plappert, M.; Tworek, J.; Hilton, J.; Nakano, R.; Hesse, C.; and Schulman, J. 2021. Training Verifiers to Solve Math Word Problems. ArXiv, abs/2110.14168.

Fan, S.; Han, P.; Shang, S.; Wang, Y.; and Sun, A. 2025. Cothink: Token-efficient reasoning via instruct models guiding reasoning models. arXiv preprint arXiv:2505.22017.

Feng, S.; Fang, G.; Ma, X.; and Wang, X. 2025. Efficient reasoning models: A survey. arXiv preprint arXiv:2504.10903.

Han, T.; Wang, Z.; Fang, C.; Zhao, S.; Ma, S.; and Chen, Z. 2025. Token-Budget-Aware LLM Reasoning. arXiv:2412.18547.

Hendrycks, D.; Burns, C.; Kadavath, S.; Arora, A.; Basart, S.; Tang, E.; Song, D.; and Steinhardt, J. 2021. Measuring Mathematical Problem Solving With the MATH Dataset. NeurIPS.

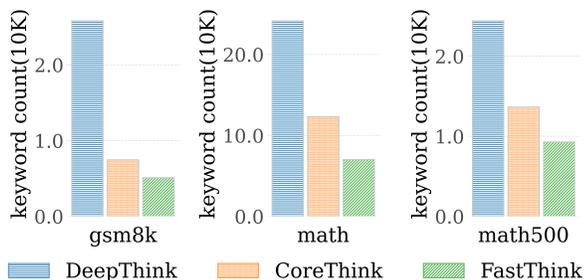
Hou, B.; Zhang, Y.; Ji, J.; Liu, Y.; Qian, K.; Andreas, J.; and Chang, S. 2025. Thinkprune: Pruning long chain-of-thought of llms via reinforcement learning. arXiv preprint arXiv:2504.01296.

Huang, S.; Wang, H.; Zhong, W.; Su, Z.; Feng, J.; Cao, B.; and Fung, Y. R. 2025a. AdaCtrl: Towards Adaptive and Controllable Reasoning via Difficulty-Aware Budgeting. arXiv:2505.18822.

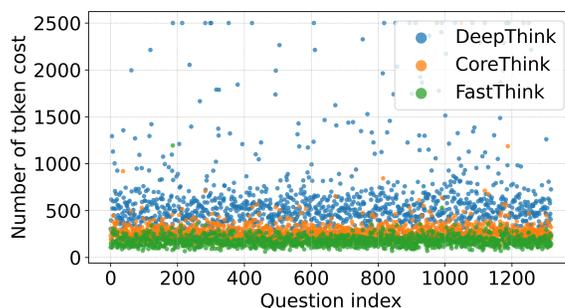
Huang, Z.; Cheng, T.; Qiu, Z.; Wang, Z.; Xu, Y.; Ponti, E. M.; and Titov, I. 2025b. Blending Supervised and Reinforcement Fine-Tuning with Prefix Sampling. arXiv:2507.01679.

Jiang, L.; Wu, X.; Huang, S.; Dong, Q.; Chi, Z.; Dong, L.; Zhang, X.; Lv, T.; Cui, L.; and Wei, F. 2025. Think only when you need with large hybrid-reasoning models. arXiv preprint arXiv:2505.14631.

Kimi. 2025. Kimi k1.5: Scaling Reinforcement Learning with LLMs. arXiv:2501.12599.



(a) 数学数据集中与推理相关的关键词频率。



(b) 在 GSM8K 上三种模式下思维长度的分布。

Figure 4: 在不同思维模式下的推理行为分析。(a) 在 GSM8K、MATH 和 MATH500 上的推理关键词频率显示更深层的模式生成更明确的推理。(b) GSM8K 上标记长度的分布突出了在 DeepThink、CoreThink 和 FastThink 之间推理深度和推断成本的明显区别。

Lee, A.; Che, E.; and Peng, T. 2025. How well do llms compress their own chain-of-thought? a token complexity approach. arXiv preprint arXiv:2503.01141.

Li, M.; Zhong, J.; Zhao, S.; Lai, Y.; Zhang, H.; Zhu, W. B.; and Zhang, K. 2025a. Think or not think: A study of explicit thinking in rule-based visual reinforcement fine-tuning. arXiv preprint arXiv:2503.16188.

Li, Z.; Dong, Q.; Ma, J.; Zhang, D.; and Sui, Z. 2025b. SelfBudgeter: Adaptive Token Allocation for Efficient LLM Reasoning. arXiv:2505.11274.

Lightman, H.; Kosaraju, V.; Burda, Y.; Edwards, H.; Baker, B.; Lee, T.; Leike, J.; Schulman, J.; Sutskever, I.; and Cobbe, K. 2023. Let's Verify Step by Step. arXiv preprint arXiv:2305.20050.

Liu, W.; Bai, Y.; Han, C.; Weng, R.; Xu, J.; Cao, X.; Wang, J.; and Cai, X. 2024. Length Desensitization in Direct Preference Optimization. arXiv:2409.06411.

Lu, J.; Yu, H.; Xu, S.; Ran, S.; Tang, G.; Wang, S.; Shan, B.; Fu, T.; Feng, H.; Tang, J.; et al. 2025. Prolonged reasoning is not all you need: Certainty-based adaptive routing for efficient llm/mlm reasoning. arXiv preprint arXiv:2505.15154.

Ma, L.; Liang, H.; Qiang, M.; Tang, L.; Ma, X.; Wong, Z. H.; Niu, J.; Shen, C.; He, R.; Cui, B.; and Zhang, W. 2025. Learning What Reinforcement Learning Can't: Interleaved Online Fine-Tuning for Hardest Questions. arXiv:2506.07527.

Shao, Z.; Wang, P.; Zhu, Q.; Xu, R.; Song, J.; Bi, X.; Zhang, H.; Zhang, M.; Li, Y. K.; Wu, Y.; and Guo, D. 2024. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models. arXiv:2402.03300.

Shen, Y.; Zhang, J.; Huang, J.; Shi, S.; Zhang, W.; Yan, J.; Wang, N.; Wang, K.; Liu, Z.; and Lian, S. 2025. Dast: Difficulty-adaptive slow-thinking for large reasoning models. arXiv preprint arXiv:2503.04472.

Sheng, G.; Zhang, C.; Ye, Z.; Wu, X.; Zhang, W.; Zhang, R.; Peng, Y.; Lin, H.; and Wu, C. 2024. Hy-

bridFlow: A Flexible and Efficient RLHF Framework. arXiv preprint arXiv: 2409.19256.

Snell, C.; Lee, J.; Xu, K.; and Kumar, A. 2024. Scaling LLM Test-Time Compute Optimally can be More Effective than Scaling Model Parameters. CoRR, abs/2408.03314.

Sui, Y.; Chuang, Y.-N.; Wang, G.; Zhang, J.; Zhang, T.; Yuan, J.; Liu, H.; Wen, A.; Zhong, S.; Chen, H.; and Hu, X. 2025. Stop Overthinking: A Survey on Efficient Reasoning for Large Language Models. arXiv:2503.16419.

Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Ichter, B.; Xia, F.; Chi, E.; Le, Q.; and Zhou, D. 2023. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. arXiv:2201.11903.

White, C.; Dooley, S.; Roberts, M.; Pal, A.; Feuer, B.; Jain, S.; Shwartz-Ziv, R.; Jain, N.; Saifullah, K.; Dey, S.; et al. 2024. LiveBench: A challenging, contamination-limited LLM benchmark. arXiv preprint arXiv:2406.19314.

Wu, H.; Yao, Y.; Liu, S.; Liu, Z.; Fu, X.; Han, X.; Li, X.; Zhen, H.-L.; Zhong, T.; and Yuan, M. 2025a. Unlocking Efficient Long-to-Short LLM Reasoning with Model Merging. arXiv:2503.20641.

Wu, T.; Xiang, C.; Wang, J. T.; Suh, G. E.; and Mittal, P. 2025b. Effectively controlling reasoning models through thinking intervention. arXiv preprint arXiv:2503.24370.

Xiang, V.; Blagden, C.; Rafailov, R.; Lile, N.; Truong, S.; Finn, C.; and Haber, N. 2025. Just Enough Thinking: Efficient Reasoning with Adaptive Length Penalties Reinforcement Learning. arXiv:2506.05256.

Zhuang, R.; Wang, B.; and Sun, S. 2025. Accelerating chain-of-thought reasoning: When goal-gradient importance meets dynamic skipping. arXiv preprint arXiv:2505.08392.