

从 "显性共同目标 "到 "隐性共同目标": 逐步学习内化协同工作

Yuntian Deng^{1,2} Yejin Choi¹, Stuart Shieber⁴ 艾伦人
工智能研究所² 滑铁卢大学
³ 华盛顿大学⁴ 哈佛大学
{yuntian, yejin}@allenai.org, shieber@seas.harvard.edu

摘要

在利用语言模型完成推理任务时,生成明确的思维链 (CoT) 步骤往往被证明是实现高精度最终输出的关键。在本文中,我们将研究是否可以教模型内化这些 CoT 步骤。为此,我们提出了一种简单而有效的内化 CoT 步骤的方法:从一个经过显式 CoT 推理训练的模型开始,我们逐步去除中间步骤并对模型进行微调。这一过程可使模型内化中间推理步骤,从而简化推理过程,同时保持高性能。我们的方法能让 GPT-2 Small 模型以高达 99% 的准确率解决 9 乘 9 的乘法运算,而标准训练无法解决超过 4 乘 4 的乘法运算。此外,我们的方法在 Mistral 7B 等更大型的语言模型上也证明有效,在 GSM8K 上的准确率超过了 50%,而且不产生任何中间步骤。

1 引言

为提高语言模型 (LM) 执行复杂推理任务的性能,一种流行的方法是思维链 (CoT) 推理,即 LM 在得出最终答案之前生成明确的中间推理步骤[14, 19]。这种方法能让模型把复杂问题分解成更简单、更容易处理的部分,从而提高最终预测的准确性。然而,这种显式推理过程的计算成本可能很高,尤其是当推理链很长的时候[6]。此外,使用显式中间步骤可能与 LM 的内在计算优势不符[12]:例如,多位数乘法对计算器来说非常容易,但对 GPT-4 来说仍然具有挑战性[20]。

在这项工作中,我们研究了将推理过程内化到模型隐藏状态中的可能性。我们提出了一种名为 "逐步内部化" 的方法,该方法从一个经过显式 CoT 推理训练的模型开始。然后,我们逐步移除中间步骤并对模型进行微调,迫使其将推理过程内部化。一旦所有中间步骤都被内化,我们就得到了一个能够进行完全隐式 CoT 推理的模型。此外,即使在模型不具备完全隐式 CoT 推理能力的情况下,这种方法仍能在保持准确性的同时缩短推理链。

我们的方法是 Deng 等人[6]提出的方法的替代方案,他们的共同目标是利用变换器的隐藏状态进行隐式推理,而不是依赖显式 CoT 标记。为了教会模型使用隐藏状态进行推理,该方法采用了一个执行显式 CoT 推理的教师模型,然后将教师的隐藏状态提炼为学生模型的隐藏状态。相比之下,我们的方法要简单有效得多。

与标准训练方法相比,我们的方法有了明显的改进。例如,使用乘法逐步内化法训练的 GPT-2 Small 模型甚至可以解决 9 乘 9 的乘法问题。

预印本。正在审查。

而不使用 CoT 的标准训练甚至连 4 乘 4 的乘法运算都很困难。此外，我们的方法还能有效地扩展到更大的模型，如 Mistral 7B 模型[10]，在 GSM8K 数据集的小学数学单词问题[5]中，我们的方法在不产生任何明确的中间步骤的情况下取得了超过 50% 的准确率，超过了更大的不带思维链推理的 GPT-4 模型，后者在提示直接生成答案时的得分只有 44%。

值得注意的是，我们的实证评估主要针对特定的推理任务，如多位数乘法和小学数学问题。虽然我们的结果显示了逐步内化法在这些情境中的潜力，而且该方法的简易性也使其适用于各种任务中的思维链方法，但我们还需要进一步的研究来探索其在更广泛的任务和更多样的 CoT 跟踪中的有效性。由于可用计算资源的限制，对其他任务的实验不在本文研究范围之内。本文旨在为这一新方法奠定基础并强调其前景，同时承认其全面普及性仍在研究之中。

我们工作的贡献如下：首先，我们介绍了 "逐步内部化" (Stepwise Internalization) --一种用于隐式 CoT 推理的简单方法。其次，我们证明了通过逐步内部化将中间隐藏状态内部化的有效性。第三，我们提供的实证结果表明，在不同的推理任务和模型规模上，使用逐步内化法训练的模型性能优越。我们的代码、数据和预训练模型可在 https://github.com/da03/Internalize_CoT_Step_by_Step 上获取。

2 背景：内隐思维推理链

隐式思维链推理 (implicit CoT, 或 ICoT) 是 Deng 等人[6]提出的一个概念，即在生成过程中，语言模型不会在词语中产生明确的中间推理步骤。它与不使用思维链推理 (No CoT) 不同，在训练过程中允许使用显式推理步骤，从而使 ICoT 模型能够从推理过程中提供的监督中学习基本的推理方法。Deng 等人[6]的主要观点是，在显式 CoT 中，中间推理步骤有两个目的：在训练过程中提供监督，以促进任务学习[14]；在推理过程中充当划板，以协助解决任务[19]。不过，后一个目的可以通过利用模型的内部状态而不是显式标记来实现。

举例来说，我们可以用语言模型来解决一个多位数乘法问题，如 12×34 。(实际输入的数字顺序颠倒为 $21 * 43$ ，以便与 Deng 等人的算法保持一致[6])。在长乘法算法中， 12×34 被分解为

$$12 \times 4 + 12 \times 30 = \underset{\text{反转} \times 84}{48} + \underset{\text{反向} \times 063}{360}$$

在显式 CoT 中，先训练模型预测这些中间步骤 $84 + 063$ ，然后再预测最终答案 804 (反向 408)。预测这些中间步骤有助于提高模型解决任务的能力。(中间步骤反转也是为了让模型更容易预测[17])。

在 "无 CoT" 和 "隐式 CoT" 两种情况下，模型都需要绕过中间步骤，直接从输入中预测出答案 408。这种方法可以大大加快长推理链的推理速度，但要以牺牲准确性为代价。

隐式 CoT 与非隐式 CoT 的主要区别在于使用中间推理步骤作为训练过程中的监督。在 Deng 等人的研究中[6]，采用了一种知识提炼方法，将显式推理提炼为隐藏状态中的隐式推理。这种方法包括训练教师模型执行显式 CoT 推理，然后将这些知识传授给学生模型，学生模型将推理过程内化到其隐藏状态中。

在目前的工作中，我们提出了一种更简单但更有效的方法，这种方法基于一种我们称之为 "逐步内化" 的课程学习，我们将在下一节详细介绍这种方法。

	Input	CoT	Output
Explicit CoT Stage 0:	2 1 × 4 3 =	8 4 + 0 6 3	= 8 0 4
Stage 1:	2 1 × 4 3 =	4 + 0 6 3	= 8 0 4
Stage 2:	2 1 × 4 3 =	+ 0 6 3	= 8 0 4
Stage 3:	2 1 × 4 3 =	0 6 3	= 8 0 4
Stage 4:	2 1 × 4 3 =	6 3	= 8 0 4
Stage 5:	2 1 × 4 3 =	3	= 8 0 4
Implicit CoT Stage 6:	2 1 × 4 3 =		= 8 0 4

图 1: 隐式思维链推理的逐步内化法。本图以解 12×34 为例, 说明了逐步内化法。训练过程包括多个阶段。在第 0 阶段, 训练模型以预测完整的思维链 (CoT) 和最终输出, 这与显式 CoT 训练相同。在第 1 阶段, 去掉第一个 CoT 标记, 然后对模型进行微调, 以预测剩余的 CoT 标记和输出。接下来的每一个阶段都会去除一个额外的 CoT 标记。到了第 6 阶段, 所有的 CoT 标记都被移除, 经过训练的模型可以根据输入直接预测输出, 从而实现隐式 CoT 推理。这种逐步移除和微调的过程可以让模型逐步内化推理步骤。

3 逐步内化

逐步内化法是一种通过在训练过程中逐步去除中间推理步骤来实现隐式思维链推理的方法。我们将输入定义为 x , 中间步骤定义为 $z = z_1, z_2, \dots, z_m$, 最终输出定义为 y 。首先使用以下损失函数训练参数为 θ 的语言模型:

$$\min_{\theta} -\log P_{\theta}(y, z_{1:m} | x),$$

其中, $z_{1:m}$ 表示中间步骤序列 z_1, z_2, \dots, z_m 。

在训练过程的每一步 t , 我们都会从中间步骤 z 中删除 (最多) $s(t)$ 个标记:

$$\min_{\theta} -\log P_{\theta}(y, z^{I+min(s(t),m):m} | x).$$

参数化 $s(t)$ 有多种方法。例如, 它可以基于损失值的阈值, 或类似于优化器中使用的学习率调度器的预定义时间表。在这项工作中, 为了简单起见, 我们使用线性时间表来删除标记:

$$s(t) = \Delta \frac{t}{T},$$

其中, T 是每个历时的总步数, 而 Δ 是一个超参数, 用于控制每个历时删除的 CoT 标记数量。(一旦 $s(t)$ 超过了实际思维链标记的数量, 所有标记都会被移除)。

在最初的实验中, 我们发现由于损失函数随时间的变化而变化, 导致训练过程不稳定。这种不稳定性主要有两个原因:

首先, 训练语言模型常用的优化器 (如 AdamW [11, 13]) 会保持对二阶梯度的估计。损失函数的突然变化 (由移除一个额外的 CoT 标记引起) 会导致二阶梯度的突然变化。为了解决这个问题, 每当移除一个额外的 CoT 标记时, 我们都会重置优化器的状态。

表 1: 数据集统计数据。标记数是基于 GPT-2 标记化器的中位数。

数据集	大小	# 输入令牌			# CoT 代币			# 输出标记		
	培训开发测试	培训开发测试			培训开发测试			培训开发测试		
4 × 4 多	808k 1k 1k	9	9	9	4646	46		9	9	9
5 × 5 多	808k 1k 1k		1111	11	7474	74		11	11	11
7 × 7 多	808k 1k 1k		1515	15	148	148	148	15	15	15
9 × 9 多	808k 1k 1k		1919	19	246	246	246	19	19	19
GSM8K	378k 0.5k 1.3k		4051	53		1921	24	2	2	2

其次，即使模型在移除 s 个标记时完全符合当前损失，但过渡到下一阶段，即移除 $s + 1$ 个标记时，损失也会显著增加，因为模型尚未针对这一新设置进行训练。为了缓解这一问题，我们引入了一种我们称之为 "移除平滑" 的技术，即在移除 $s(t)$ 的原始代币数上添加一个小的随机偏移量，从而：

$$s(t)^* = s(t) + o,$$

其中， o 是一个支持非负整数 $\mathbb{Z}_{\geq 0}$ 的随机变量，其分布由另一个超参数 λ 参数化：

$$P(o) \propto \exp(-\lambda o).$$

当 $\lambda = \infty$ 时， $o = 0$ ，我们恢复了没有移除平滑的版本。然而，当 $\lambda < \infty$ 时，模型会被训练成以很小的概率在第 t 步移除超过 $s(t)$ 个标记，这有助于平滑过渡到移除 $s(t) + 1$ 个标记的下一阶段，减少损失函数中的突然跳变。

图 1 展示了 "逐步内化" 方法的高级理念。训练过程由多个阶段组成，在每个阶段中，模型通过删除 CoT 中的标记，逐步学会内化推理步骤，最终实现隐式 CoT 推理。

4 实验装置

4.1 数据

我们按照 Deng 等人[6]的方法，在两个推理任务上评估了我们提出的逐步内化方法：多位数乘法和小数乘法推理。

多位数乘法。 我们使用了 BIG-bench 中最具挑战性的两项算术任务 [3]:Deng 等人[6] 所描述的 4 乘 4 乘法和 5 乘 5 乘法。鉴于逐步内化法在这些任务中的有效性，我们将评估范围扩大到 7 乘 7 和 9 乘 9 的乘法运算。乘法任务的复杂度会随着位数的增加而显著增加，因为程序长度会随着位数的增加而呈二次曲线增长 [7]。我们使用 Deng 等人[6]的脚本和设置为主要实验生成合成训练数据¹。

小学数学我们使用的是 GSM8K 数据集[5]，以及 Deng 等人提供的增强训练数据[6]。表 1 提供了详细的数据集统计信息。

4.2 基线和模型

我们将我们的方法与以下基线进行了比较：

- **无 CoT**：直接训练模型，无思维链监督。
- **显式 CoT**：用明确的思维链推理对模型进行微调或提示[14]。我们在 GPT 3.5 和 GPT-4 中使用了 5 次提示，但在其他模型中使用了完全微调。
- **ICoT-KD**：Deng 等人提出的通过知识提炼的隐式思维链方法[6]。

¹根据 Deng 等人的研究[6]， K -by- K 乘法只考虑 K 位数，而不考虑更低的数位。

表 2: 乘法任务的结果。ICoT-KD: 通过知识提炼实现的内隐 CoT, 数字取自 Deng 等人 [6]。ICoT-SI: 通过逐步内化实现的隐式 CoT (本研究)。准确度 (表中的 Acc) 衡量产生最终答案的精确匹配准确度。速度衡量的是在批量为 1 的情况下, 推理过程中每秒的示例数量, 并以相应的无 CoT 模型的速度进行归一化。除了 ICoT-SI 实验所基于的 GPT2-Small 模型外, 为了便于比较, 我们还提供了一组无 CoT 和显式 CoT 模型。

一组不同尺寸模型的 CoT 基线。[†]: 5 发提示, 而不是微调。

型号	4 × 4		5 × 5		7 × 7		9 × 9	
	Acc	速度	Acc	速度	Acc	速度	Acc	速度
小型 GPT-2 (117M)								
明确的 CoT	1.00	0.17	1.00	0.14	1.00	0.12	1.00	0.09
无 CoT	0.29	1.00	0.01	1.00	0.00	1.00	0.00	1.00
ICoT-KD	0.97	0.67	0.10	0.71	-	-	-	-
ICoT-SI	1.00	1.02	0.95	1.00	0.95	1.00	0.99	1.00
MathGLM-100M								
无 CoT	0.80	1.00	0.56	1.00	-	-	-	-
MathGLM-500M								
无 CoT	0.90	1.00	0.60	1.00	-	-	-	-
MathGLM-2B								
无 CoT	0.95	1.00	0.90	1.00	-	-	-	-
GPT-3.^{5†}								
明确的 CoT	0.43	0.10	0.05	0.07	0.00	0.15	0.00	0.11
无 CoT	0.02	1.00	0.00	1.00	0.00	1.00	0.00	1.00
GPT-4†								
明确的 CoT	0.77	0.14	0.44	0.14	0.03	0.09	0.00	0.07
无 CoT	0.04	1.00	0.00	1.00	0.00	1.00	0.00	1.00

表 3: 各种方法在 GSM8K 上的精度。[†]: 5 次提示, 而非微调。

模型	GPT-2 小型	GPT-2 介质	Phi-3 3.8B	Mistral 7B	GPT-3.5 [†]	GPT-4 [†]
明确的 CoT	0.41	0.44	0.74	0.68	0.62	0.91
无 CoT	0.13	0.17	0.28	0.38	0.03	0.44
ICoT-KD	0.20	0.22	-	-	-	-
ICoT-SI	0.30	0.35	0.31	0.51	-	-

我们提出的方法, 即通过逐步内化实现隐式思维链, 被称为 ICoT-SI。为了验证我们的方法在不同模型规模下的有效性, 我们使用了预训练模型 GPT-2 [16]、Phi-3 3.8B [1] 和 Mistral-7B [10]。

4.3 评估

由于隐式思维链方法的前提是接近无思维链的速度和显式思维链的准确性, 因此我们使用了两个主要的评估指标: 首先, 我们评估每种方法在生成最终输出的各自任务上的准确性。其次, 我们将每种方法的推理速度与无 CoT 基准进行比较。对于 ICoT-KD, 我们直接采用 Deng 等人的数据[6]。然而, 由于硬件差异, 当无法获得 ICoT-KD 的速度数据时, 我们会重新计算相对于 No CoT 的速度。

5 成果

表 2 列出了主要结果, 我们将逐步内部化与各种基线进行了比较。

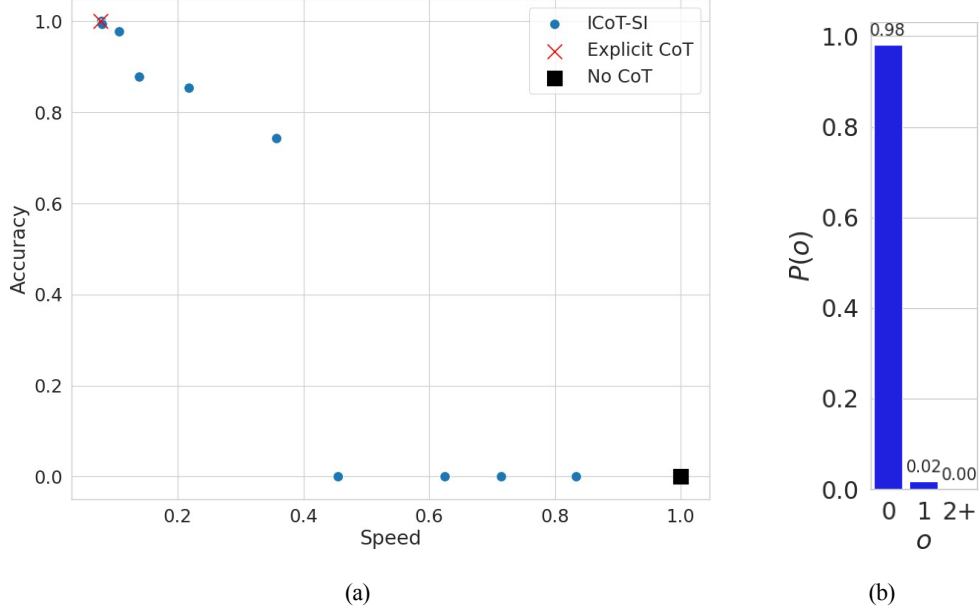


图 2: (a) ICoT-SI 的速度与精度权衡。该图说明了在使用 GPT-2 Small 的 11×11 乘法任务中，逐步内部化方法 (ICoT-SI) 在速度和准确性之间的权衡。随着越来越多的 CoT 标记被移除和内部化，准确度会降低，而速度会提高。在两个极端，我们的方法既能恢复显式 CoT（准确率高，但速度慢），也能恢复无 CoT（速度非常快，但准确率为 0）。请注意，在该图中，我们删除了被其他点“支配”的点（即存在另一个速度更快、精度更高的点）。(b) $\lambda = 4$ 的移除平滑中随机移除偏移 o 的分布。分布主要集中在 $o = 0$ ，概率为 0.98，而 $o \geq 1$ 的概率仅为 0.02。尽管如此，移除平滑仍被证明是有效的，这一点已在烧蚀研究中得到证实。

逐步内化是有效的。与其他不输出中间步骤的方法相比，逐步内化法 (ICoT-SI) 被证明是非常有效的。例如，ICoT-SI 可使 GPT-2 小模型以 0.99 的精度解决 9×9 的乘法问题，而无 CoT 方法甚至连 4×4 的乘法问题都无法解决。此外，ICoT-SI 还优于通过知识蒸馏实现的隐式 CoT (ICoT-KD)；ICoT-KD 无法使用 GPT-2 Small 模型解决 5×5 乘法问题，而 ICoT-SI 却能解决高达 9×9 的乘法问题。此外，虽然 ICoT-KD 由于额外的仿真器模型而比 No CoT 稍慢，但 ICoT-SI 的速度与 No CoT 相同。²

与现有文献相比，ICoT-SI 也很有竞争力。例如，在模型大小相似的情况下，MathGLM-100M [20] 只能以 0.56 的精度求解 5×5 的乘法运算。即使有 20 亿个参数，MathGLM-2B 也能以 0.90 的精度求解 5×5 的乘法运算。虽然另一项相关工作 [17] 能够训练一个 GPT-2 Small 模型，以解决高达 14×14 的乘法运算，但该工作中提出的方法仅限于算术任务，而 ICoT-SI 则更具通用性。

ICoT-SI 以一种通用的方式实现了 CoT 推理的内部化，使其适用于算术以外的任务，如小学数学问题。例如，在 GSM8K 数据集上，ICoT-SI 使未使用任何中间步骤的模型达到了最新的准确度。它对 Mistral-7B 模型进行了微调，使精度超过了 0.50，而 GPT-4 甚至只能达到 0.44 而不使用中间步骤。

逐步内化法在准确性上落后于显式 CoT，但速度更快。就精度而言，隐式 CoT 方法仍然落后于显式 CoT。例如，经过微调的 Mistral-7B 模型在 GSM8K 上的显式 CoT 精确度为 0.68，而 ICoT-SI 的精确度仅为 0.51。但是

² 由于硬件速度的随机性，表 2 中 ICoT-SI 的速度并不总是 1.00。

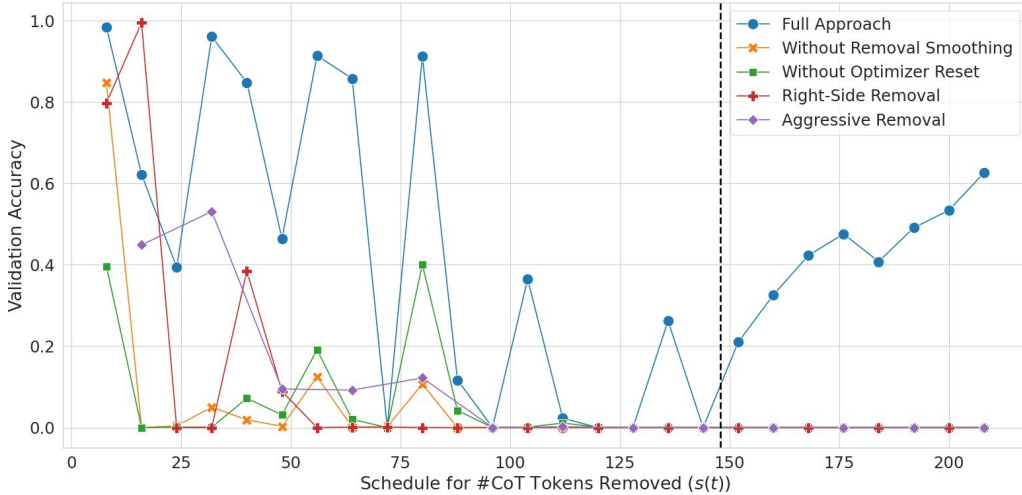


图 3: 训练期间各种删减的准确率。该图显示了在使用 GPT-2 Small 进行 7×7 乘法任务的训练过程中, 验证准确率与可能移除的 CoT 标记数量的函数关系。黑色垂直虚线表示计划删除所有 CoT 标记的时间点。曲线比较了以下变体: "完全方法"、"无移除平滑" (不使用 $\lambda = 4$ 的移除平滑, 即 $\lambda = \infty$)、"无优化器重置" (移除标记后不重置优化器状态)、"右侧移除" (CoT 标记从末端而非起点移除) 和 "激进移除" (每个历时移除 16 个而非 8 个 CoT 标记)。所有这些变体的性能都低于完整方法。更多详情, 请参见第 6.2 节。

隐式 CoT 方法具有显著的速度优势。例如, 在 9×9 乘法任务中, ICoT-SI 的准确度与显式 CoT 相当, 但推理速度却快 11 倍。

总之, 我们的研究结果表明, 逐步内化法是一种有效的隐式 CoT 推理方法, 能在准确性和速度之间做出令人信服的权衡。因此, 对于需要高性能和低延迟的任务来说, 它是一种有价值的方法。

6 分析

6.1 精度与速度的权衡

ICoT-SI 的一个显著优势是, 它可以通过内化不同数量的 CoT 令牌来权衡准确性和速度。一个极端是, 当没有令牌被内化时, ICoT-SI 可以恢复显式 CoT 性能。在另一个极端, 当所有代币都被内化时, 我们就能实现隐式 CoT, 其性能通常比直接训练无 CoT 模型要好得多。

即使 ICoT-SI 由于模型容量的限制而不能完全成功, 例如在更具挑战性的任务中, 它无法内部化所有 CoT 步骤, 我们仍然可以利用中间检查点来实现精度和速度之间的权衡。例如, 如图 2a 所示, 在使用 GPT-2 Small 的 11×11 乘法任务中, 即使模型无法内化所有 CoT 步骤, ICoT-SI 仍能在内化部分 CoT 令牌的情况下, 以四倍于显式 CoT 的速度实现超过 0.7 的准确率。

这条权衡曲线说明了 ICoT-SI 在平衡计算效率和模型性能方面的灵活性。通过调整内化 CoT 标记的数量, 用户可以根据具体应用的要求, 优化提高准确性或加快推理速度。

6.2 消融研究

图 3 显示了在 7×7 乘法任务的训练过程中去除的 CoT 标记数量与验证准确率的关系。该图将完整方法与几种

消融变体。即使是完整方法，曲线也会出现波动，在训练过程中，验证准确率曾一度短暂下降到零，但最终又恢复了。然而，当准确率下降时，消融变体并不能完全恢复。

移除平滑化。如第 3 节所述，当损失函数因移除更多的 CoT 标记而发生变化时，在移除的标记数中添加一个小的随机偏移量 o 至关重要。正如第 3 节所介绍的， o 的分布是由超参数 λ 参数化的。我们在整个工作中使用 $\lambda = 4$ ，从而得到图 2b 所示的分布。在这种分布中，98% 的情况下 $o = 0$ ，但约有 2% 的情况下会删除一个或多个额外标记。如图 3 所示，"无移除平滑"曲线在 $s(t) = 50$ 左右准确度降为零后无法恢复，而完整方法的准确度要好得多。

重置优化器稳定训练的另一项重要技术是在移除更多代币时重置优化器。这可以避免对二阶导数的大量估计，并稳定训练。在图 3 中，"未重置优化器"曲线在 100 步左右降为零，且不再恢复，这说明了在训练过程中重置优化器的重要性。

移除侧。在我们的主要实验中，CoT 标记从开始部分（左侧）开始移除。如图 3 中的"右侧移除"曲线所示，从右侧移除 CoT 标记的效果要差得多。我们怀疑这是因为在开头内化标记比在结尾内化标记更容易。结尾处的 CoT 标记依赖于前面的标记，因此在 CoT 结束和最终答案开始（只有几个位置）之间内化这些标记更具挑战性。与此相反，将标记内化到开头则可以将标记分布到整个输入中。

每个历元删除的词块数量。每次删除的词块数 (Δ) 对训练的稳定性 and 速度有很大影响。在主要实验中，我们使用了 $\Delta = 8$ ，即每个 epoch 删除 8 个标记。 Δ 值越大，训练速度越快，但有可能无法收敛，因为模型可能跟不上损失函数的快速变化。例如，当使用 $\Delta = 16$ 时，训练无法收敛，如图 3 中的"积极去除"曲线所示。相反， Δ 值越小，训练越容易成功，但速度越慢。未来的工作可以探索基于损失值的自适应 Δ 计划，以更有效地平衡速度和稳定性。

7 相关工作

无 CoT 方法。有几篇文献的研究重点是训练语言模型来解决算术任务，而不输出中间步骤。MathGLM [20] 证明，只要有足够的训练数据，包括低位数和高位数算术任务演示，20 亿参数的 LM 就能在不输出任何中间步骤的情况下解决多位数算术任务。与这项工作相比，"逐步内化"法在解决多位数乘法问题时，用更小的模型就能达到更高的准确率，这很可能是由于在训练过程中利用了思维链监督。Shen 等人[17]的另一项著名研究表明，通过混合低位数和高位数乘法演示，即使是 GPT-2 Small 也能学习到 14 位数乘法。然而，逐步内化不需要专门准备混合任务难度的训练数据。此外，从理论上讲，分步内化法适用于任何具有 CoT 推理步骤的推理任务，它在小学数学问题上的有效性就证明了这一点。

Pfau 等人[15]的研究也与此相关，他们的研究表明，转换器语言模型可以使用填充标记来替代 CoT 标记进行推理。他们的研究表明，使用这些填充标记进行推理可以提高语言模型的表达能力。我们的方法有可能与他们的方法相结合，以解决更具挑战性的任务。

内化 CoT。我们的工作与 Deng 等人[6] (ICoT-KD) 的工作密切相关，后者引入了隐式 CoT 推理任务。ICoT-KD 允许在训练过程中使用 CoT，但不允许在生成过程中使用 CoT，它通过知识提炼将推理步骤内化到隐藏状态中来实现这一点。与 ICoT-KD 相比，逐步内化有三个优势：首先，它不需要教师模型，因此实施起来更简单。其次，ICoT-KD 将推理内化为单一的状态"列"（与最终输入位置相对应）、

逐步内部化允许模型在所有输入位置进行内部推理。最后，与 ICoT-KD 相比，逐步内部化的准确性更高。

我们的工作还与语境蒸馏（Context Distillation）[18] 有关，后者训练模型在有刮板和没有刮板的情况下产生相同的输出。逐步内化的每个阶段都可以看作是一种语境蒸馏形式，其中一个 CoT 标记被蒸馏到模型的内部状态中。逐步内部化将情境蒸馏扩展到课程学习环境中。

另一项相关工作是 Searchformer [12]，它首先训练一个变换器来模仿 A* 搜索，然后在采样的较短搜索轨迹上对其进行微调。这样，模型就能使用比训练时提供的步骤更少的步骤来执行搜索。Searchformer 依靠采样来找到更短的踪迹，而 Stepwise Internalization 则通过移除 CoT 标记来强制模型内部化步骤。

8 局限性

训练成本。由于在移除每组 CoT 标记时都需要进行微调，拟议方法的一个局限性是训练成本较高。正如第 6.2 节中所讨论的，移除 CoT 标记的速度过快会导致不收敛。因此，CoT 链越长，训练时间就越长。对于像 N 位乘法这样的任务，推理链的长度会随着 N 的增加呈指数增长，因此随着 N 的增加，训练的成本也会越来越高。

不稳定性。我们观察到的另一个实际问题是使用激进的 Δ 值进行训练时的不稳定性。例如，附录 B 中的图 4 显示了模型无法从精度下降中恢复的情况。使用较低的 Δ 值通常会使得训练更稳定，但代价是训练时间更长。如 Hu 等人[9]所建议的那样，通过重新开始训练，及早识别并解决不稳定动态，可能是一种有价值的改进。

可解释性。与现有的 No CoT 和隐式 CoT 训练工作类似，使用我们的方法训练出来的模型失去了可解释的中间步骤。不过，使用探测技术也许可以解释这些模型的内部隐藏状态[2, 8]。此外，结合隐式和显式 CoT 训练可以让用户在可解释性和延迟之间做出选择，从而根据未来任务的要求提供灵活性。

准确性。毫无疑问，与我们的隐式 CoT 方法相比，显式 CoT 仍然能达到更高的精度。不过，我们的方法可以在延迟和准确性之间做出权衡。即使在没有中间步骤就无法完全解决的任务上，如 11×11 乘法，它也能保持合理的准确性，同时比显式 CoT 快几倍。此外，我们的研究结果还证明了利用隐藏状态进行推理的潜力：尽管 GPT-2 Small 模型只有 12 层，远远少于 9×9 乘法的 CoT 中的推理步骤数，但它也能被训练成求解 9×9 乘法。当扩展到拥有千亿参数和多达百层的大型模型时，如 GPT-3 [4]，它们就有可能在没有显式 CoT 步骤的情况下解决更具挑战性的推理任务。

9 结论和未来工作

在这项工作中，我们介绍了逐步内部化（Stepwise Internalization），这是一种在语言模型中实现隐式思维链推理的新方法。通过逐步移除中间 CoT 标记并对模型进行微调，我们逐步实现了推理步骤的内部化。与现有方法相比，我们的方法有了显著的改进，在使用 GPT-2 Small 进行高达 9×9 的乘法运算时达到了很高的准确率，在 GSM8K 上的表现优于 GPT-4，同时不输出任何中间推理步骤。与显式 CoT 方法相比，我们的方法在保持类似精度的同时，速度最多可提高 11 倍。

在未来的工作中，当模型将每个推理步骤内部化时，对内部过程进行探测可以深入了解学习机制。此外，开发一种结合了隐式和显式 CoT 推理的混合模式方法，有可能实现两全其美，在准确性、延迟和基于用户偏好的可解释性之间取得平衡。另一个有前途的方向是将逐步内部化扩展到更大的模型和更广泛的训练/再训练设置中，这样可以进一步提高其在更广泛的推理任务中的有效性。

致谢和资金披露

这项工作得到了美国国家科学基金会 (NSF) DMS-2134012 号基金和美国国家航空研究局 (ONR) N00014-24-1-2207 号基金的支持。我们还要感谢哈佛大学 FAS Research Computing 提供的计算资源。

参考资料

- [1] Marah Abdin、Sam Ade Jacobs、Ammar Ahmad Awan、Jyoti Aneja、Ahmed Awadallah、Hany Awadalla、Nguyen Bach、Amit Bahree、Arash Bakhtiari、Harkirat Behl、Alon Benham、Misha Bilenko、Johan Bjorck、Sébastien Bubeck、Martin Cai、Caio César Teodoro Mendes、Weizhu Chen、Vishrav Chaudhary、Parul Chopra、Allie Del Giorno、Gustavo de Rosa、Matthew Dixon、Ronen Eldan、Dan Iter、Amit Garg、Abhishek Goswami、Suriya Gunasekar、Emman Haider、Junheng Hao、Russell J. Hewett、Jamie Huynh、Mojan Javaheripi、Xin Jin、Piero Kauffmann、Nikos Karampatziakis、Dongwoo Kim、Mahoud Khademi、Lev Kurilenko、James R. Lee、Yin Tat Lee、Yuanzhi Li、Chen Liang、Weishung Liu、Eric Lin、Zeqi Lin、Piyush Madan、Arindam Mitra、Hardik Modi、Anh Nguyen、Brandon Norick、Barun Patra、Daniel Perez-Becker、Thomas Portet、Reid Pryzant、Heyang Qin、Marko Radmilac、Corby Rosset、Sambudha Roy、Olatunji Ruwase、Olli Saarikivi、Amin Saied、Adil Salim、Michael Santacrose、Shital Shah、Ning Shang、Hiteshi Sharma、Xia Song、Masahiro Tanaka、Xin Wang、Rachel Ward、Guanhua Wang、Philipp Witte、Michael Wyatt、Can Xu、徐佳航、Sonali Yadav、杨帆、杨子怡、于东晗、张成瑞东、张茜、张建文、张丽娜、张毅、张悦、张云安和周希仁。Phi-3 技术报告：手机上的高性能语言模型，2024 年。
- [2] Yonatan Belinkov. *深度学习中的内部语言表征：机器翻译和语音识别分析*。博士论文，麻省理工学院，2018 年。
- [3] 大板凳作者。超越模仿游戏：量化和推断语言模型的能力。《机器学习研究论文集》，2023 年。ISSN 2835-8856。URL <https://openreview.net/forum?id=uyTL5Bvosj>.
- [4] Tom B. Brown、Benjamin Mann、Nick Ryder、Melanie Subbiah、Jared Kaplan、Prfulla Dhariwal、Arvind Neelakantan、Pranav Shyam、Girish Sastry、Amanda Askell、Sandhini Agarwal、Ariel Herbert-Voss、Gretchen Krueger、Tom Henighan、Rewon Child、Aditya Ramesh、Daniel M. Ziegler、Jeff Wu、Clemens Winter、Christopher Hesse、Mark Chen、Eric Sigler、Mateusz Litwin、Scott Gray。Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 语言模型是少量学习者，2020 年。
- [5] 卡尔-科布、维尼特-科萨拉朱、穆罕默德-巴伐利亚、马克-陈、希乌-俊、卢卡斯-凯泽、马蒂亚斯-普拉珀特、杰里-特沃雷克、雅各布-希尔顿、中野礼一郎、克里斯托弗-黑塞和约翰-舒尔曼。训练验证器解决数学文字问题，2021 年。
- [6] 邓云天、基兰-普拉萨德、罗兰-费尔南德斯、保罗-斯摩伦斯基、维什拉夫-乔杜里和斯图尔特-希伯。通过知识提炼进行隐式思维链推理》，2023 年。
- [7] Nouha Dziri、Ximing Lu、Melanie Sclar、Xiang Lorraine Li、Liwei Jiang、Bill Yuchen Lin、Sean Welleck、Peter West、Chandra Bhagavatula、Ronan Le Bras 等。信仰与命运：变压器对构成性的限制。《神经信息处理系统进展》，36，2024。
- [8] 约翰-休伊特和珀西-梁设计和解释带有控制任务的探测器，2019 年。
- [9] Michael Y. Hu, Angelica Chen, Naomi Saphra, and Kyunghyun Cho. 训练动态的潜在状态模型》，2024 年。
- [10] Albert Q. Jiang、Alexandre Sablayrolles、Arthur Mensch、Chris Bamford、Devendra Singh Chaplot、Diego de las Casas、Florian Bressand、Gianna Lengyel、Guillaume Lample、Lucile Saulnier、Lélio Renard Lavaud、Marie-Anne Lachaux、Pierre Stock、Teven Le Scao、Thibaut Lavril、Thomas Wang、Timothée Lacroix 和 William El Sayed。米斯特拉尔 7b，2023 年。
- [11] Diederik P. Kingma 和 Jimmy Ba. 亚当：随机优化方法》，2017 年。

- [12] Lucas Lehnert、Sainbayar Sukhbaatar、DiJia Su、Qinqing Zheng、Paul Mcvay、Michael Rabbat 和 Yuandong Tian。超越 a^* ：通过搜索动力学引导更好地规划变压器，2024。
- [13] 伊利亚-洛希洛夫和弗兰克-胡特解耦权重衰减正则化学习表征国际会议，2019 年。URL <https://openreview.net/forum?id=Bkg6RiCqY7>。
- [14] Maxwell Nye、Anders Johan Andreassen、Guy Gur-Ari、Henryk Michalewski、Jacob Austin、David Bieber、David Dohan、Aitor Lewkowycz、Maarten Bosma、David Luan、Charles Sutton 和 Augustus Odena。展示您的作品用于语言模型中间计算的 Scratchpads，2021 年。
- [15] Jacob Pfau、William Merrill 和 Samuel R. Bowman。让我们逐点思考：转换器语言模型中的隐藏计算》，2024 年。
- [16] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 语言模型是无监督的多任务学习者。 *OpenAI blog*, 1(8):9, 2019.
- [17] 沈若琪、塞巴斯蒂安-布贝克、罗南-埃尔丹、李银达、李远志、张毅。变压器算术的位置描述问题，2023 年。
- [18] 查理-斯内尔 (Charlie Snell)、丹-克莱因 (Dan Klein) 和钟瑞琪 (Ruiqi Zhong)。通过提炼语境来学习，2022。
- [19] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. 大型语言模型中的思维链提示推理。见 Alice H. Oh、Alek Agarwal、Danielle Belgrave 和 Kyunghyun Cho 编辑的《*神经信息处理系统进展*》，2022 年。URL https://openreview.net/forum?id=_VjQlMeSB_J。
- [20] 杨震、丁明、吕青松、蒋志环、何泽海、郭玉怡、白金凤、唐杰。Gpt 无需计算器即可解决数学问题》，2023 年。

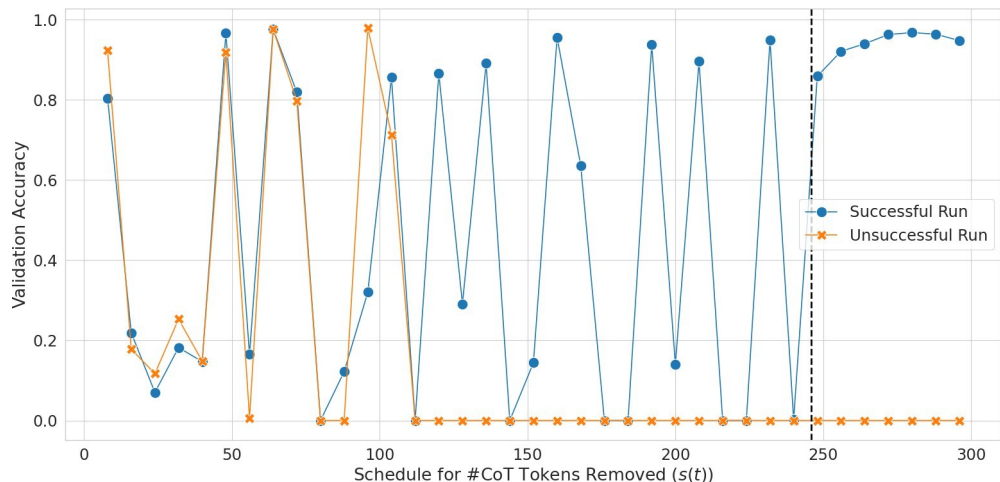


图 4: 两种不同随机种子在训练过程中的验证精度。该图显示了在使用 GPT-2 Small 和 $\Delta = 8$ 进行 9×9 乘法任务的训练过程中, 验证准确率与可能移除的 CoT 标记数量的函数关系。黑色垂直虚线表示移除所有 CoT 标记的位置。

A 超参数

在所有实验中, 我们使用 AdamW 优化器 [13], 默认设置为 $\lambda = 4$, 有效批量大小为 32。对于 Phi-3 3.8B 和 Mistral 7B, 我们使用的批量大小为 16, 梯度累积为 2。对于乘法任务, 我们使用的学习率为 5×10^{-5} , $\Delta = 8$ 。对于 GSM8K, 我们使用的学习率为 5×10^{-5} , $\Delta = 1$ 的学习率, 对于 GPT-2 Small 和 GPT-2 Medium, 使用 5×10^{-5} 的学习率和 $\Delta = 1$ 的学习率。对于 Phi-3 3.8B 和 Mistral 7B, 训练率为 1×10^{-5} , $\Delta = 8$, 精度为 bfloat16。此外, 对于 GSM8K, 我们只考虑训练 150 个或更少的标记序列, 并删除所有 CoT。当计划删除 39 个或更多标记时。所有实验都在一台配备 80GB GPU 内存的 H100 上运行, 最多可运行 200 个历元或 24 小时, 以先到者为准。

B 强力清除的稳定性问题

我们发现, 使用激进的移除计划 (即较大的 Δ 值) 有时会导致不稳定的训练动态。例如, 图 4 显示了在除随机种子外的相同配置下进行的两次不同运行。其中一次运行在移除所有 CoT 标记后最终解决了任务, 而另一次运行在移除所有 CoT 标记后未能解决任务。

C 附加实验

保留位置 ID。随着 CoT 标记的删除, 最终输出开始的位置也会发生变化。我们尝试了一种位置 ID 保持不变的变体, 即在移除 CoT 标记后直接使用下一个标记的位置 ID。虽然这种方法在训练过程中更加稳定, 但其性能与当前方法类似。为了简单起见, 我们在主要实验中没有使用这种变体。

可供选择的 CoT 格式。对于同一个问题, 不同的有效推理路径可以得出正确的最终答案。我们探索了在乘法问题中使用二叉树格式的 CoT 链。这种格式将 N 位数乘法分解为 N 位数乘 1 位数的乘法问题序列, 使用和运算符合并结果, 并继续合并直到计算出最终和。这个程序的描述长度较短, 可能更容易被转换器学习 [7]。不过, 它的性能与目前的方法类似: 在使用 GPT-2 Small 进行 9×9 乘法时, 它的准确率达到了 0.95, 而在 11×11 乘法时则失败了。