Trend Report: Chain Compression in reasoning model

> CoT, or Chain-of-Thought, is a technique employed by large language models (LLMs) to generate a series of intermediate reasoning steps for tackling complex tasks, such as arithmetic, common sense, and symbolic reasoning (Wei et al., 2022, Chain-of-Thought Prompting Elicits Reasoning in Large Language Models). Although CoT significantly enhances the reasoning capabilities of models, the generation of long chains leads to high reasoning costs and computational resource demands. As model sizes continue to expand, the need for efficient reasoning becomes increasingly urgent. **Research on chain compression in reasoning models aims to reduce these costs by shortening the reasoning chains while striving to maintain or only minimally impact model performance.**

Definition

Chain compression refers to the process of shortening the reasoning paths generated by reasoning models while ensuring that essential reasoning steps are preserved or that model performance does not significantly decline. This can be achieved through the following methods:

- Refining the reasoning process into a more compact form.
- Efficiently storing and recalling reasoning steps using the model's internal states.
- Employing summarization or abstraction techniques to reduce the verbosity of the chain.
- Dynamically adjusting the length of the chain based on the complexity of the task.
- Training the model to generate shorter chains through reinforcement learning or other optimization techniques.

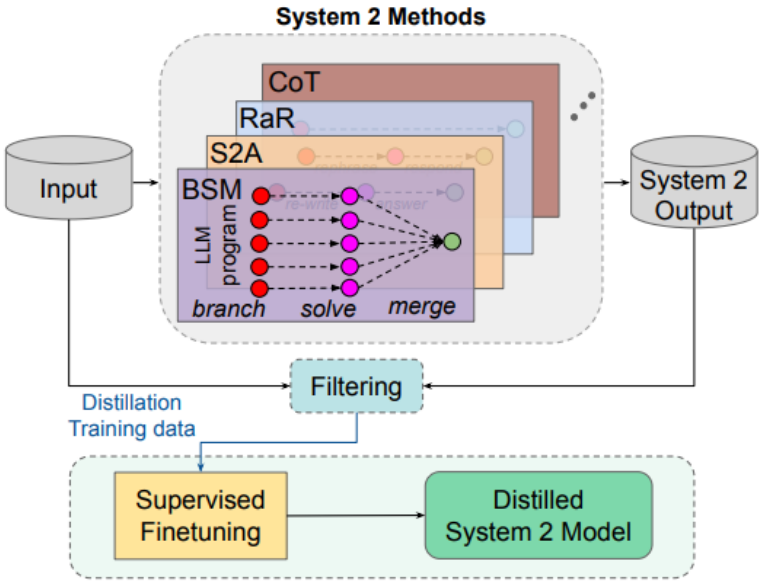
Method

训练阶段显式压缩

[Arxiv'7 24] Distilling System 2 into System 1

- Meta的工作

- Abs: 研究直接把reasoning model kd到base model



- **整体方案**
  a. **System 1与System 2的区分**
     - **System 1**：直接基于输入生成最终输出，不产生中间推理步骤（如标准LLM的单步生成），响应速度快但复杂任务表现有限
     - **System 2**：生成中间推理标记（如思维链、分支-解决-合并等），通过多步提示或搜索提升结果质量，但计算成本高、延迟显著

  b. **蒸馏流程**
     - **步骤1：生成高质量响应**
       使用System 2方法（如Rephrase and Respond、System 2 Attention等）处理未标注输入数据，生成包含中间步骤的响应
     - **步骤2：自监督筛选数据**
       通过**Self-Consistency**过滤噪声数据：
       - **输出自洽性**：对同一输入多次采样System 2响应，保留多数一致的结果，丢弃分歧样本
       - **输入扰动自洽性**：轻微修改输入（如调整多选题顺序），若System 2输出不稳定则丢弃样本
     - **步骤3：微调System 1**
       用筛选后的数据（输入-输出对）微调System 1模型，使其直接模仿System 2的高质量结果，**无需生成中间推理标记**

- Comments
  - 很早期的工作了，在o1之前就有了

[NIPS'24] Can Language Models Learn to Skip Steps?

- 邱组的工作
- **Abs**：通过多步训练的方式逐步减少推理路径

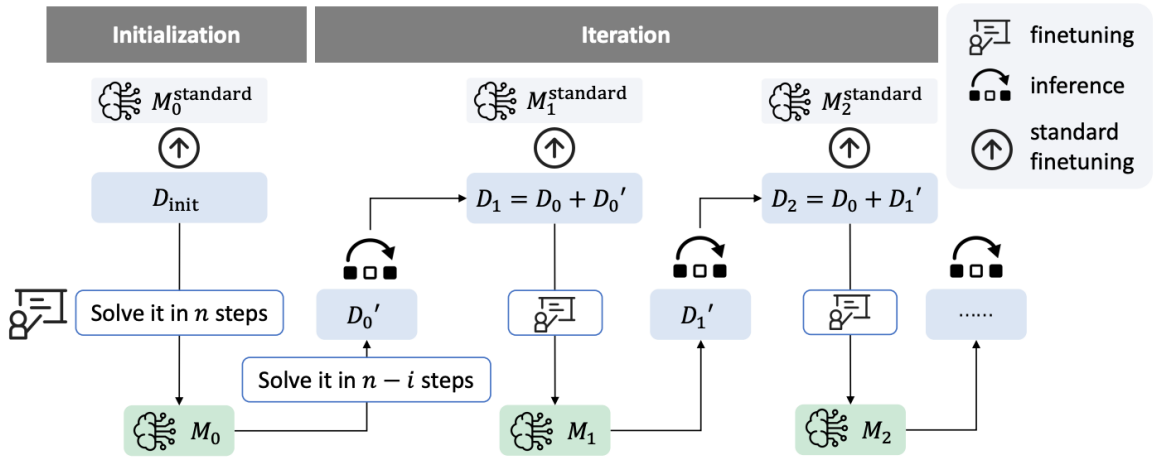


Figure 2: Overall framework. The initialization phase aims to equip the model with the ability to reason according to a specified number of steps. During iterations, each cycle produces a mixed dataset D_i, which is used to train a standard model to evaluate the model's step-skipping capabilities.

- **整体方案**
  - 初始化阶段：使用包含完整推理步骤的数据集训练模型（如代数类比的详细推导），建立基础推理能力。
  - 迭代优化阶段
    - 跳步路径生成：控制训练环境，引导模型生成更简短的推理路径（如省略多位数加法中的重复进位步骤）。
    - 混合数据集构建：将已验证准确性的跳步路径与完整步骤数据混合，形成扩展训练集。
    - 迭代改进：通过多轮训练，逐步提升模型识别冗余步骤的能力，减少生成标记量的同时保持准确性。
- **技术细节**
  - 自监督数据筛选：通过验证跳步路径的准确性（如对比完整步骤结果），自动过滤错误跳步样本。
  - 动态训练策略：采用梯度控制技术，平衡跳步带来的效率提升与潜在错误风险。
  - 任务适配性：针对不同任务（如方向推理、代数类比）设计特定的跳步规则，例如在方向推理中跳过重复的方向判断步骤。
- **实验**
  - Bench

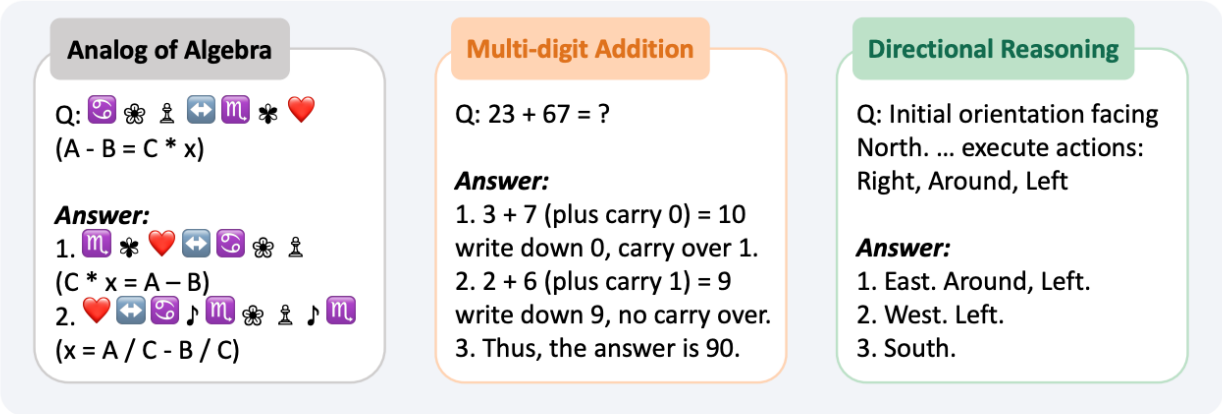


Figure 3: Illustrations of three different tasks. Each question is accompanied by a comprehensive detailed step-by-step solution.

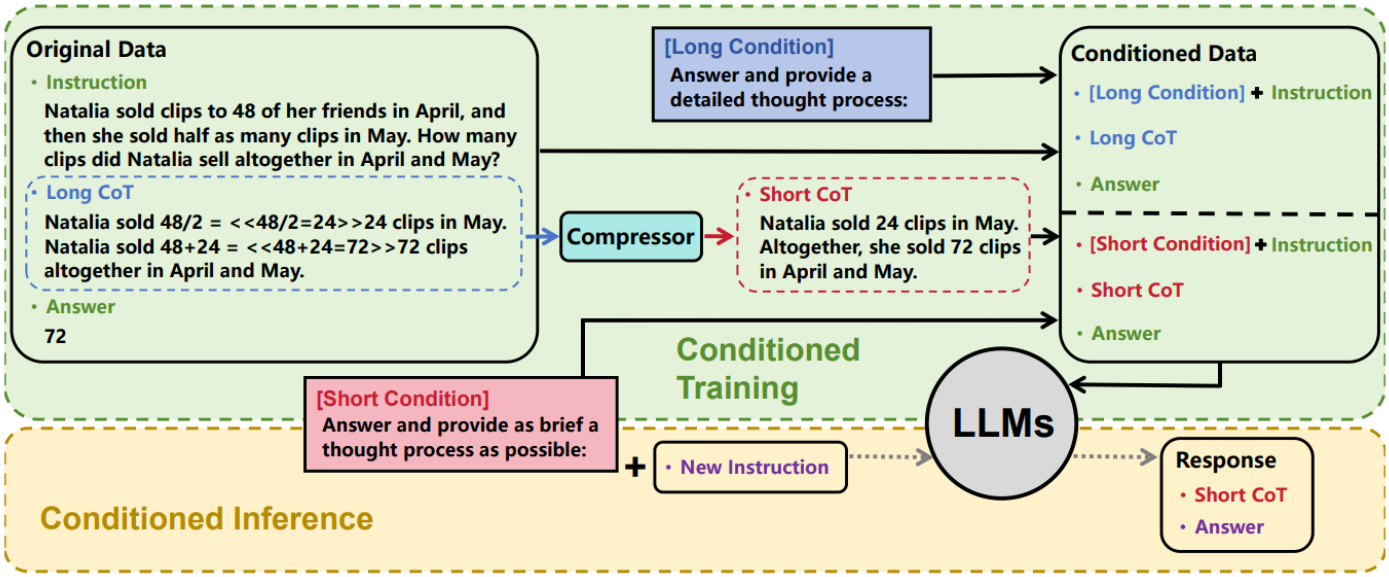
- Result

	Analogy of Algebra		Multi-digit Addition		Directional Reasoning	
	$i=1$	$i=2$	$i=1$	$i=2$	$i=1$	$i=2$
# Skipping	5,308	4,159	2,844	2,175	2,071	2,049
Step Consistency	100.00	99.19	100.00	100.00	86.24	39.19
Answer Accuracy	8.14	2.77	98.35	82.58	85.47	29.62
Average Step	2.33	1.81	1.90	1.38	6.14	6.66

- Comments
  - 早期工作，有一些insight
  - Base model还是llama2，参考价值比较小
  - 需要根据不同的任务设计，局限性比较大
  - 没有在核心bench上实验
  - 靠逐步训练解决问题，比较trivial

[AAAI'25] C3oT: Generating Shorter Chain-of-Thought without Compromising Effectiveness

- 贝壳网的工作
- **Abs**：用GPT造short cot的数据，用special token区分两种数据再放在一起训练



- **整体方案**

  a. 压缩器（Compressor）

  - 作用：将原始的较长 CoT 压缩为更短的 CoT，同时保留关键信息和推理的可解释性。
  - 实现方式：
    - 文章使用 GPT-4 作为压缩器。
    - GPT-4 对原始 CoT 进行处理，生成一个精简版本，确保保留所有必要的推理步骤和逻辑。
    - 例如，一个复杂的算术问题可能包含冗长的中间步骤，压缩器会删除多余的描述，仅保留核心推理内容。

  b. 条件训练方法（Conditioned Training Method）

  - 作用：训练大型语言模型同时适应较长的 CoT 和较短的 CoT，学习它们之间的对应关系。
  - 实现方式：
    - 在训练数据中包含成对的较长 CoT 和较短 CoT。
    - 使用不同的prompt tokens来区分两种 CoT。例如：
      - 较长 CoT 以 [LONG] 开头。
      - 较短 CoT 以 [SHORT] 开头。
    - 通过这种方式，模型能够根据提示标记学习生成不同长度的 CoT，并理解较长和较短 CoT 之间的逻辑联系。

  c. 条件推理方法（Conditioned Inference Method）

  - 作用：在推理阶段，引导模型生成更短的 CoT，同时利用从较长 CoT 中学习到的推理能力。
  - 实现方式：
    - 在推理时，使用与较短 CoT 相关的初始提示标记（例如 [SHORT]）。

- 模型根据提示生成精简的 CoT，而无需依赖冗长的推理过程。
- 这种方法确保推理效率提高，同时保持准确性。

- **实验**

Model Size	Method	Arithmetic				Commonsense			
		GSM8K		MathQA		ECQA		StrategyQA	
		Acc	Compression Rate	Acc	Compression Rate	Acc	Compression Rate	Acc	Compression Rate
7B	Short CoT	31.01	58.63	46.16	29.53	61.93	53.41	67.59	37.99
	Long CoT	**37.38**	0	**51.46**	0	63.96	0	69.66	0
	Implicit-CoT	11.16	100	14.62	100	21.14	100	30.01	100
	C3oT	**36.92**	56.67	50.35	27.39	**69.38**	51.55	**72.41**	42.04
13B	Short CoT	42.46	59.52	52.97	29.85	66.79	55.27	74.83	47.79
	Long CoT	**48.07**	0	**56.21**	0	68.92	0	**76.21**	0
	Implicit-CoT	14.36	100	17.00	100	23.54	100	35.77	100
	C3oT	**47.10**	57.78	**56.62**	31.04	**71.93**	55.28	**76.55**	44.56

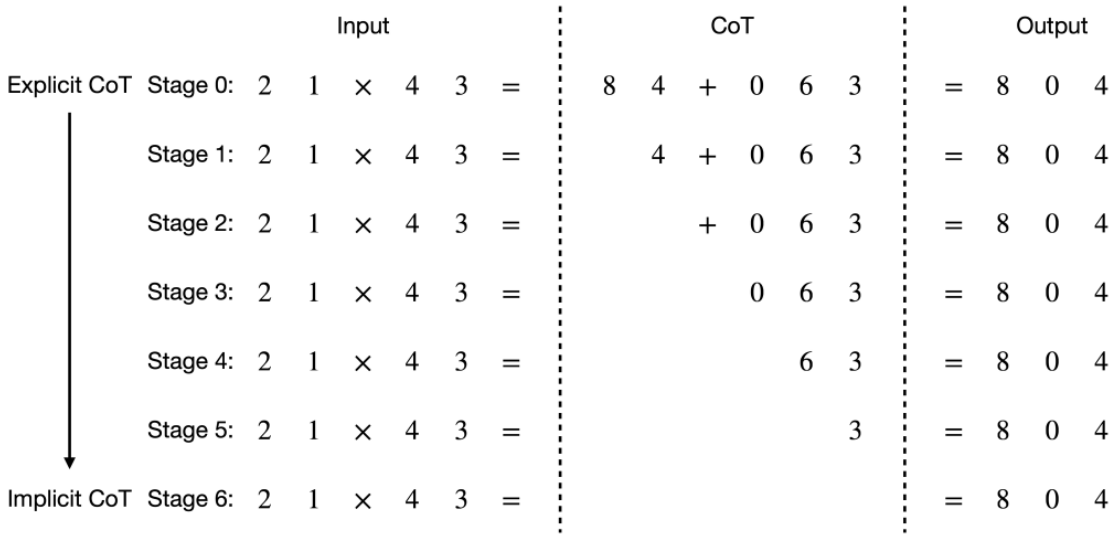
Table 1: The Accuracy (%) and Compression Rate (%) performance of the proposed C3oT and baselines. The **bold** scores denote the best performance, as well as performances within 1% of the best.

- Comments
  - 有在GSM8K上的实验了，看着效果还不错
  - 但是只有在7B、13B级别的结果，但对学术界来说也没有办法
  - 方法上比较简单，但GPT造的short cot数据不一定优质

隐藏状态压缩

[Arxiv'5 24] From Explicit CoT to Implicit CoT: Learning to Internalize CoT Step by Step

- Yejin Choi组的工作
- Abs: 从显式CoT逐步训练减少token数，最后全部内化到模型的hidden state中



- **整体方案**
  a. **具体步骤**
    i. **显式CoT训练阶段**：
      - 初始阶段，模型被训练生成完整的显式CoT中间步骤（如计算过程）和最终答案。
      - 损失函数：最小化联合概率 $-\log P_\theta(y, z_{1:m} \mid x)$，其中 $z_{1:m}$ 为中间步骤序列。
    ii. **逐步移除中间步骤**：
      - **线性调度**：按预设的线性计划逐步移除CoT token。例如，每训练一个阶段（epoch）移除固定数量的token。
      - **移除平滑（Removal Smoothing）**：为避免训练不稳定，引入随机偏移量 o，以一定概率提前移除更多token，平滑过渡到下一阶段。
    iii. **优化器重置**：
      - 每次移除新token时，重置优化器状态（如AdamW的二阶梯度估计），避免梯度突变导致的训练不稳定。
    iv. **隐式CoT推理**：
      - 当所有中间步骤被移除后，模型仅需输入问题并直接输出答案，中间推理完全内化到隐藏状态中。
  b. **关键技术细节**
    - **移除方向**：优先从中间步骤的**左侧（起始位置）**开始移除，实验表明左侧移除更有效。
    - **训练稳定性**：
      - **移除平滑**：通过概率分布控制额外移除的token数量（如 $P(o) \propto \exp(-\lambda o)$），减少训练突变。
      - **优化器重置**：防止梯度估计偏差积累。
- 实验
  - bench：GSM8K
  - result：

Model	GPT-2 Small	GPT-2 Medium	Phi-3 3.8B	Mistral 7B	GPT-3.5†	GPT-4†
Explicit CoT	0.41	0.44	0.74	0.68	0.62	0.91
No 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	-	-

- Comment
  - 看结果一般，效果下降很多，模型也都是比较弱的

- 比较早的工作，和邱组的工作比较像，但是邱的工作是显示地压缩，减少无效的token，最终还是要有一条interpretable的显示推理链，Yejin Choi的工作是从一个方向逐步内化到hidden中去，最终没有推理链了

[Arxiv'12 24] Compressed Chain of Thought: Efficient Reasoning Through Dense Representations

- JHU的工作
- Abs：用隐空间的有密集语义的token代表压缩的推理过程，通过压缩推理链的隐藏状态（hidden states），保留推理的关键信息。

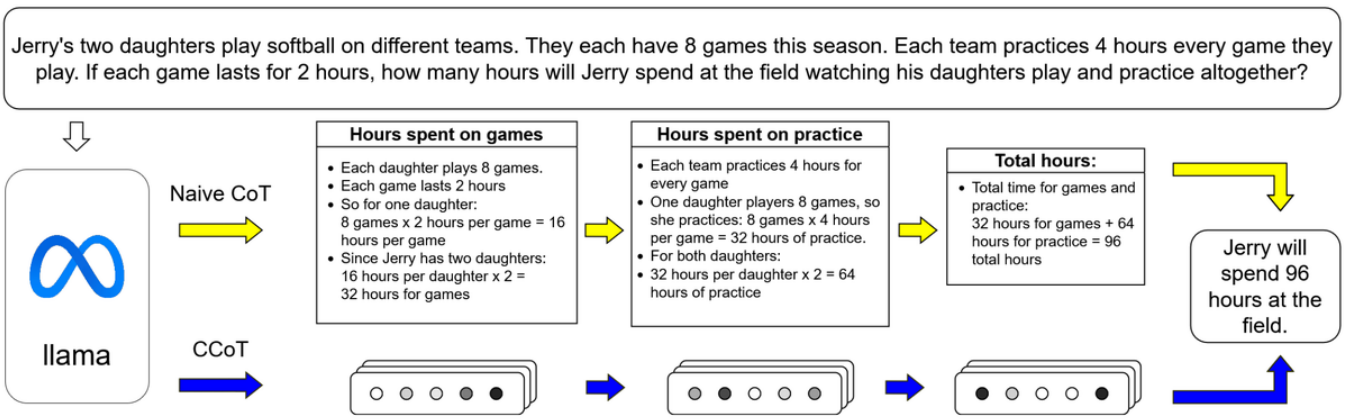


Figure 1. Two approaches to step by step reasoning. Chain of Thought (CoT) prompting reasons via discrete language tokens, leading to long sequences that incur significant generation costs. In contrast Compressed Chain of Thought (CCOT) elicits reasoning with a short sequence of continuous embeddings, allowing for much greater throughput.

- **整体方案**
  a. 两个模块：
    - **CCoT 模块**：负责生成 contemplation tokens。
    - **DECODE 模块**：根据查询和生成的令牌解码最终答案。
  b. 训练数据和压缩比率

    假设有一个训练实例，包括查询（query, $w_{1:n}$）、完整的推理链（reasoning chain, $t_{1:m}$）和答案（answer, $a_{1:o}$）。CCoT引入压缩比率 r（$0 < r < 1$），控制推理链的压缩程度，其中 $k = \lceil r \cdot m \rceil$ 是生成的沉思令牌数量。

    - $r = 1$：使用完整的推理链，等同于传统CoT。
    - $r = 0$：不生成任何沉思令牌，直接输出答案。

  c. CCOT模块

CCoT模块（参数为 φ）的目标是生成 k 个沉思令牌 $\tilde{z}_{1:k}$，这些令牌是完整推理链隐藏状态 $\hat{t}_{1:m}$ 的压缩表示。具体步骤如下：

1. **预计算隐藏状态**：
   - 将查询、推理链和答案拼接（$[w_{1:n}; t_{1:m}; a_{1:o}]$），通过预训练模型（参数 θ）计算隐藏状态：

$$[\hat{w}_{1:n}; \hat{t}_{1:m}; \hat{a}_{1:o}] = \mathrm{ATTN}_\theta([\bar{w}_{1:n}; \bar{t}_{1:m}; \bar{a}_{1:o}])$$

   - 使用一个评分器（scorer，从Qin et al., 2024借用）从 $\hat{t}_{1:m}$ 中选择 k 个代表性隐藏状态 $z_{1:k}$ 作为“金标”（gold labels）。
2. **生成沉思令牌**：
   - 直接生成 k 个沉思令牌，而不是先生成完整推理链再压缩。
   - 输入采用前一个沉思令牌的中间层隐藏状态（第 l 层，z_{i-1}^l），以自回归方式生成下一个沉思令牌 $\tilde{z}_i$。
   - 初始输入 z_0 是查询最后一个令牌的隐藏状态。

d. DECODE模块

DECODE模块（参数为 ψ）负责基于查询和生成的沉思令牌解码答案：

1. **生成沉思令牌**：
   - 使用训练好的 CCOT_φ 自回归生成 $z^*_{1:k}$。
2. **解码答案**：
   - 将查询 $w_{1:n}$ 和沉思令牌 $z^*_{1:k}$ 拼接，计算隐藏状态，然后生成答案 $a_{1:o}$。
   - 使用交叉熵损失微调 ψ：

$$\mathrm{LOSS}_\psi(a_{1:o}) = -\sum_{i=2}^{o} \log p(a_i \mid a_{1:i-1})$$

3. **优化**：
   - 在微调 ψ 时，部分解冻 φ 的后期层参数（l 层之后），以利用下游任务的反馈信号。

e. 推理过程

推理时，给定查询 w：

1. 计算查询的隐藏状态 $\hat{w}$。
2. 使用 CCOT_φ 自回归生成沉思令牌，直到二元分类器 END_ψ 预测结束（最大生成 $h = 200r$ 个令牌）。
3. 使用 DECODE_ψ 基于 $[\hat{w}; z]$ 生成答案。

与传统CoT的关键区别在于，CCoT使用第 l 层隐藏状态直接生成连续的沉思令牌，而非通过语言模型头部解码离散令牌。

- 实验
  - Bench：GSM8K
  - Result:

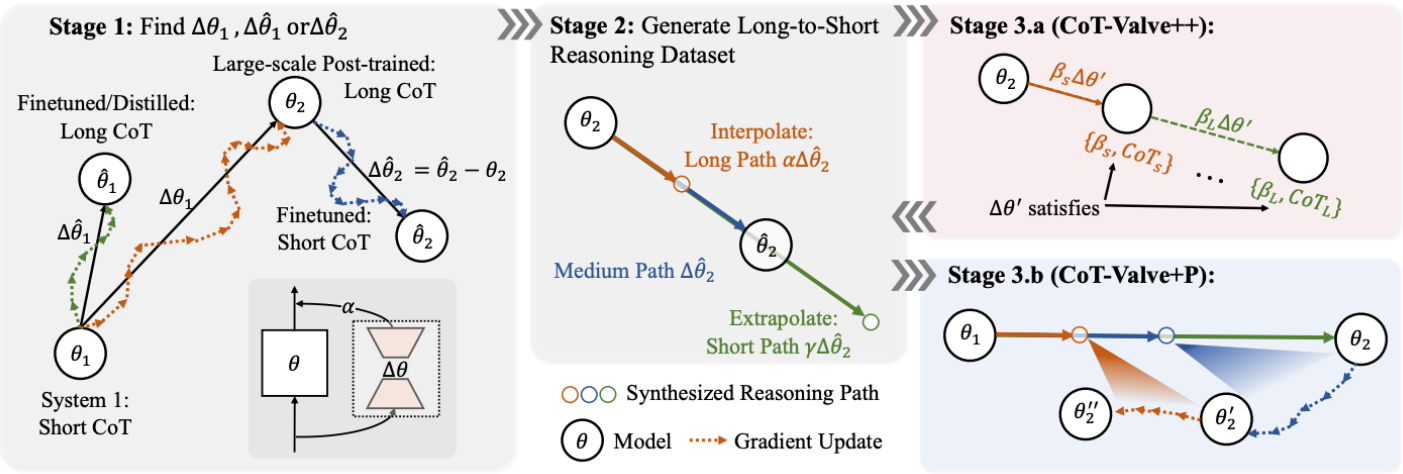
Format	$1/r$	Acc. (EM)	Decode Time
CCOT	∞	0.089	0.33
CCOT	20x	0.151	0.49
CCOT	10x	0.179	0.78
CCOT	1x	0.315	8.10
PAUSE	20x	0.092	0.35
PAUSE	10x	0.099	0.37

- Comments
  - 比起Yejin Choi的工作效果更好，也更合理一些（毕竟隔了半年了）
  - 用teaching forcing的方法训练，肯定会有一些典型的问题，比如Exposure Bias，而且很难泛化到其他任务，包括同领域的任务，感觉应该测一下在AIME的结果

动态长度控制

[Arxiv'2 25] CoT-Valve: Length-Compressible Chain-of-Thought Tuning

- NUS的工作
- Abs：用 LoRA在参数空间中引入 Δθ。通过少量额外参数实现对推理长度的调节。



- **方法：**
  - 参数空间方向控制：
    - 在模型参数空间中找到方向 Δθ，通过调整其强度（α 参数）控制生成链的长度：

$$\theta_{\text{new}} = \theta + \alpha\Delta\theta$$

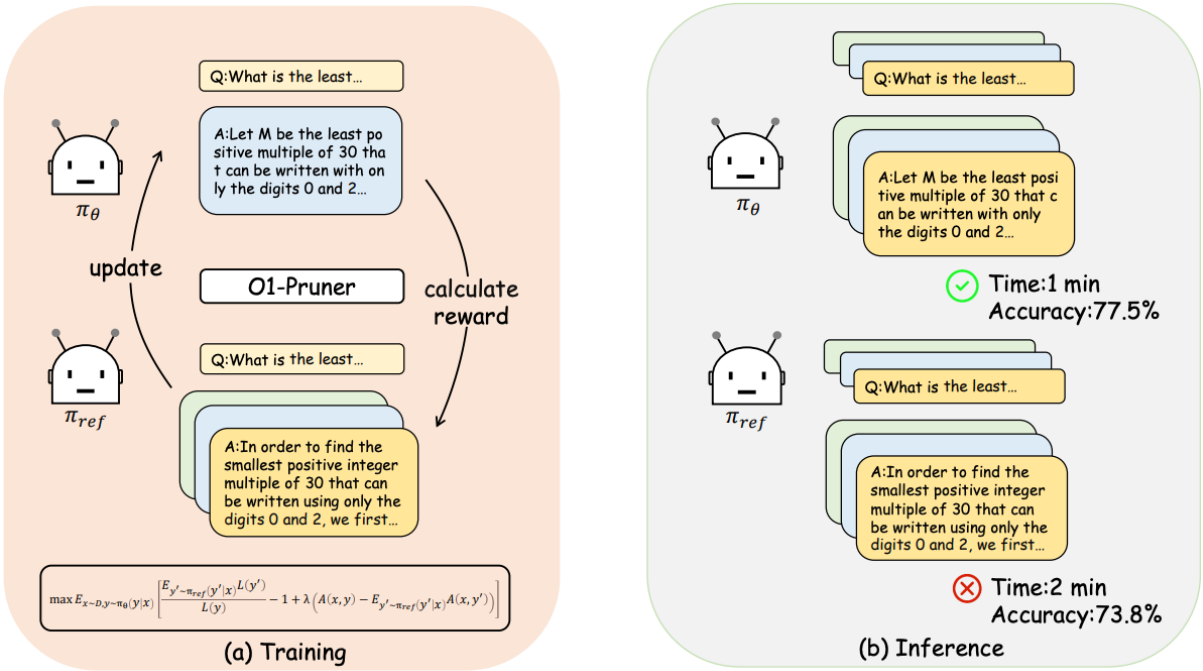
    - 使用 LoRA作为轻量级分支，实现高效调整。
    - 插值（0 < α < 1）：平滑过渡生成长短链；外推（α > 1）：生成更短的链。
  - MixChain 数据集：
    - 为每个问题构建从长到短的多条推理链。
    - 通过模型间参数差异（如基础模型与推理模型）生成数据，无需人工标注。
  - 改进策略：
    - CoT-Valve++：引入归一化参数 β，显式约束不同链长的生成能力。
    - 渐进式压缩：分阶段逐步缩短链长，类似模型剪枝中的迭代优化。
- 实验
  - 场景：
    - 长链到短链：在 QwQ-32B-Preview 上，将 GSM8K 的推理链从 741 token 压缩到 225 token（精度从 95.07% 降至 94.92%）。
    - 短链到长链：在 LLaMA-3.1-8B 上，从短 CoT 蒸馏出长 CoT 能力。
    - 短-长-短：先将 Qwen-2.5-32B-Instruct 训练为长链模型，再压缩回短链。
  - 结果：
    - 压缩效果：在 AIME 数据集上，QwQ-32B-Preview 的 token 数从 6827 降至 4629，仅多错一个答案。

    - 性能提升：短链有时优于长链，例如 LLaMA-3.2-1B 在 GSM8K 上从 52.7%（759 token）提升到 55.5%（267 token）。
    - 优于基线：相比提示控制和其他压缩方法（如 Overthink、O1-Pruner），CoT-Valve 实现更短链和更高精度。
- 发现
  a. 长链并非总是最佳，CoT-Valve 生成的短链更高效。
  b. 过长或过短的链对**小模型**学习不利，中等长度链更优。
- Comment
    - 近期的工作，其实就是训lora，但是实验量很充分（AIME和GSM8K上都做了，ablation也很充分），故事讲的不错，要投nlp会的同学可以参考一下，看篇幅应该是投acl的
    - 但感觉有点weird，这个工作是把长cot压缩成短cot，这件事的意义和价值好像不大。

强化学习压缩

[Arxiv'1 25] O1-Pruner: Length-Harmonizing Fine-Tuning for O1-Like Reasoning Pruning

- 中山的工作
- Abs：设计一个长度协调奖励函数，用RL控制模型生成更短但仍准确的推理序列



- 方法：
    - 优化目标：

O1-Pruner 的优化目标是：

- **减少推理长度**：相对于参考模型 π_{ref}，缩短输出序列长度 $L(\mathbf{y})$。
- **保持或提升准确性**：通过准确性函数 $A(\mathbf{x}, \mathbf{y})$（返回 0 或 1 表示答案是否正确）作为约束。

数学形式为：

$$\max \mathbb{E}_{\mathbf{x}\sim\mathcal{D}}\left[\mathbb{E}_{\mathbf{y}\sim\pi_\theta(\mathbf{y}|\mathbf{x}),\mathbf{y}'\sim\pi_{\text{ref}}(\mathbf{y}|\mathbf{x})}\left(\frac{L(\mathbf{y}')}{L(\mathbf{y})}-1\right)\right]$$

约束条件：

$$\mathbb{E}_{\mathbf{x}\sim\mathcal{D},\mathbf{y}\sim\pi_\theta(\mathbf{y}|\mathbf{x})}[A(\mathbf{x},\mathbf{y})] \geq \mathbb{E}_{\mathbf{x}\sim\mathcal{D},\mathbf{y}'\sim\pi_{\text{ref}}(\mathbf{y}|\mathbf{x})}[A(\mathbf{x},\mathbf{y}')]$$

为了求解，将约束转化为惩罚项，引入权重 $\lambda \geq 0$，优化目标变为：

$$\max \mathbb{E}_{\mathbf{x}\sim\mathcal{D},\mathbf{y}\sim\pi_\theta(\mathbf{y}|\mathbf{x})}\left[\frac{\hat{L}_{\text{ref}}(\mathbf{x})}{L(\mathbf{y})}-1+\lambda(A(\mathbf{x},\mathbf{y})-\hat{A}_{\text{ref}}(\mathbf{x}))\right]$$

其中：

- $\hat{L}_{\text{ref}}(\mathbf{x})$ 和 $\hat{A}_{\text{ref}}(\mathbf{x})$ 是参考模型的平均长度和准确性，通过对 π_{ref} 采样 K 次计算：

$$\hat{L}_{\text{ref}}(\mathbf{x})=\frac{1}{K}\sum_{i=1}^{K}L(\mathbf{y}_i'),\quad \hat{A}_{\text{ref}}(\mathbf{x})=\frac{1}{K}\sum_{i=1}^{K}A(\mathbf{x},\mathbf{y}_i')$$

- 长度协调函数

定义奖励函数：

$$R_{LH}(\mathbf{x},\mathbf{y})=\frac{\hat{L}_{\text{ref}}(\mathbf{x})}{L(\mathbf{y})}-1+\lambda(A(\mathbf{x},\mathbf{y})-\hat{A}_{\text{ref}}(\mathbf{x}))$$

- **长度奖励项** $\frac{\hat{L}_{\text{ref}}(\mathbf{x})}{L(\mathbf{y})}-1$：鼓励生成短序列。当 $L(\mathbf{y})$ 比 $\hat{L}_{\text{ref}}(\mathbf{x})$ 短时，奖励为正；反之为负。
- **准确性奖励项** $\lambda(A(\mathbf{x},\mathbf{y})-\hat{A}_{\text{ref}}(\mathbf{x}))$：平衡长度与准确性。对于简单问题，正确答案的奖励较小，模型倾向于缩短输出；对于复杂问题，正确答案的奖励较高，模型优先确保准确性。

- 训练策略

由于 $L(\mathbf{y})$ 和 $A(\mathbf{x}, \mathbf{y})$ 不可导，采用策略梯度方法优化。考虑到频繁采样 π_θ 会增加训练复杂度，作者引入**离线策略（off-policy）训练**：

- 从 π_{ref} 采样数据，而不是当前模型 π_θ。
- 使用 PPO 风格的损失函数：

$$L^{\text{LH}}(\theta; \mathbf{x}, \mathbf{y}) = -\mathbb{E}_{\mathbf{x}\sim\mathcal{D}, \mathbf{y}\sim\pi_{\text{ref}}(\mathbf{y}|\mathbf{x})} \left[\min\left(r(\theta)R_{LH}(\mathbf{x}, \mathbf{y}), \text{clip}(r(\theta), 1-\epsilon, 1+\epsilon)R_{LH}(\mathbf{x}, \mathbf{y})\right)\right]$$

其中 $r(\theta) = \frac{\pi_\theta(\mathbf{y}|\mathbf{x})}{\pi_{\text{ref}}(\mathbf{y}|\mathbf{x})}$ 是概率比，clip 函数限制更新幅度以提高稳定性。

- 算法流程

1. 初始化参考模型 $\pi_{\text{ref}} = \pi_\theta$。
2. 对每个问题 $\mathbf{x}^i$，从 π_{ref} 采样 K 个解，计算 $\hat{L}_{\text{ref}}$ 和 $\hat{A}_{\text{ref}}$。
3. 随机选择 m 个解，基于 L^{LH} 更新模型参数 θ。
4. 迭代直到收敛。

- 实验
  - bench：MATH、GSM8K、GaoKao
  - result：

Model	MATH			GSM8K			GaoKao			*AVERAGE*		
	Acc	Length	AES	Acc	Length	AES	Acc	Length	AES	Acc	Length	AES
Marco-o1-7B *(full fine-tune)*												
Baseline	73.8	1156	0	89.2	530	0	57.1	1112	0	73.4	932	0
Fast-solving Prompt	71.0	1113	0.15	81.7	447	0.41	57.1	1062	0.04	69.9	874	0.20
SFT	73.6	1076	0.08	89.9	497	0.09	56.3	1066	0.08	73.3	880	0.08
DPO	71.8	761	0.42	88.6	410	0.25	56.6	780	0.32	72.3	650	0.33
O1-Pruner	**77.5**	**657**	**0.58**	**91.4**	**343**	**0.43**	**61.6**	**664**	**0.64**	**76.8**	**554**	**0.55**
QwQ-32B-Preview *(freeze fine-tune last 48 layers)*												
Baseline	90.6	2191	0	95.1	777	0	79.0	2183	0	88.2	1717	0
Fast-solving Prompt	90.2	1763	0.21	95.8	561	0.30	78.4	1911	0.15	88.1	1411	0.22
SFT	90.4	2031	0.08	95.7	717	0.10	79.5	2112	0.05	88.5	1620	0.08
DPO	**91.7**	1999	0.12	95.3	704	0.10	79.7	2021	0.10	88.9	1575	0.11
O1-Pruner	91.0	**1385**	**0.38**	**96.5**	**534**	**0.36**	**80.3**	**1446**	**0.39**	**89.3**	**1121**	**0.38**

- Comment
  - 用reward function做rl来控制推理链的长度是最直观的做法，但是训好并不容易。

[Arxiv'1 25] Kimi k1.5: Scaling Reinforcement Learning with LLMs

- Kimi1.5的tech report中Chain Compression的节选
- Abs: tech report中尝试了四个compress的方案

- 方法：
  - 四个方案：
    - **Model Merging**

      Model merging has been found to be useful in maintaining generalization ability. We also discovered its effectiveness in improving token efficiency when merging a long-cot model and a short-cot model. This approach combines a long-cot model with a shorter model to obtain a new one without training. Specifically, **we merge the two models by simply averaging their weights.**

    - **Shortest Rejection Sampling**

      We observed that our model generates responses with a large length variation for the same problem. Based on this, we designed the Shortest Rejection Sampling method. **This method samples the same question n times (in our experiments, n=8) and selects the shortest correct response for supervised fine-tuning.**

    - **DPO**

      Similar with Shortest Rejection Sampling, we utilize the Long CoT model to generate multiple response samples. **The shortest correct solution** is selected as the **positive sample**, while **longer responses** are treated as **negative samples**, including both wrong longer responses and correct longer responses (1.5 times longer than the chosen positive sample). **These positive-negative pairs form the pairwise preference data used for DPO training.**

    - **Long2short RL**

      After a standard RL training phase, **we select a model that offers the best balance between performance and token efficiency to serve as the base model**, and conduct a separate long2short RL training phase. In this second phase, we apply the **length penalty** introduced in Section 2.3.3, and **significantly reduce the maximum rollout length to further penalize responses that exceed the desired length while possibly correct.**
- 实验：
  - Bench：MATH500，AIME2024
  - Result：

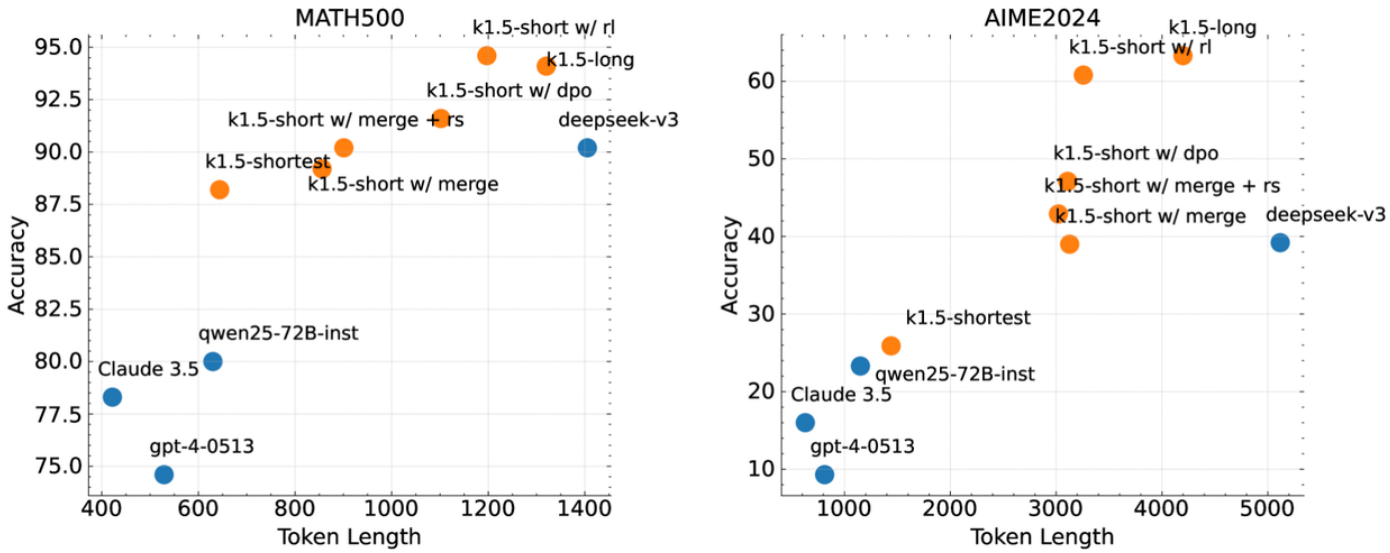


Figure 7: Long2Short Performance. All the k1.5 series demonstrate better token efficiency compared to other models.

- Comment
  - Tech report的含金量无需赘述，最有参考价值的一篇
  - 从他们对比的方法上看， RL无疑是最优解，但是从method上看好像RL的方案的初始模型和其他模型不一样

- Multi-agent framework to social reasoning