SELFCAD: PROTECTING YOUR EFFICIENT REASONING CAPABILITIES VIA SELF-CAUTIOUS INSERTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Large reasoning models (LRMs) are increasingly utilized in AI systems due to their accuracy, efficiency, and transparency, as their reasoning traces enable interpretation of model outputs by users and auditors. However, publishing these traces poses new risks: adversaries could use them to replicate efficient LRMs or create proxy models for malicious attacks, raising both copyright and security concerns. Current defenses either detect distillation post-violation or suppress transparency by masking or rewriting traces, which are impractical in real-world applications. We propose a defense framework that preserves reasoning traces while preventing effective distillation with minimal additional cost. Our analysis identifies that the number of self-cautious sentences significantly impacts model efficiency and accuracy: too many result in redundancy, while too few reduce accuracy. Based on this, we introduce SelfCAD (Self-Cautious Anti-Distillation), a lightweight method that manipulates self-cautious components after reasoning traces are generated. SelfCAD preserves trace clarity for users and auditors while significantly hindering the efficiency and accuracy of distilled models. Experiments on Llama and Qwen demonstrate that distilled models incur higher inference costs and lower accuracy, particularly for Qwen-1.5B, whose token length is $9.1\times$ longer on GSM8K, demonstrating the effectiveness of our method.

1 INTRODUCTION

Large reasoning models (LRMs) are widely used in many AI applications, including search engines, chatbots, educational platforms, and productivity tools (Touvron et al., 2023; OpenAI, 2023; Nam et al., 2024; Dam et al., 2024). Training them requires substantial investment in data, computation, and expertise. Yet even when commercial APIs hide states and logits, unauthorized developers can still collect high-quality responses for distillation (Xu et al., 2024b; Wang et al., 2022). This risks intellectual property infringement because both parameters and outputs carry economic and scientific value. The problem is especially severe for reasoning models, whose responses contain richer information than those of standard LLMs. Unauthorized distillation also creates privacy and security risks (Cui et al., 2025), including membership inference and model extraction attacks (Tramèr et al., 2016; Liang et al., 2024a). Thus, despite black-box deployment, commercial LLMs remain vulnerable, raising growing concerns (Sweney & Milmo, 2025) and threatening community openness.

Simply suppressing outputs is infeasible, as societal expectations and regulations increasingly require transparent and interpretable AI systems, such as the EU AI Act [1] and GDPR [2]. Existing defenses are also limited. Post-hoc methods such as watermarking and fingerprinting (Zhang & Koushanfar, 2024; Xu et al., 2024a; Liang et al., 2024b) can help trace misuse, but add training or inference costs and cannot prevent distillation. Although Savani et al. (2025) explored proactive defenses, their method requires costly teacher fine-tuning to rewrite reasoning trajectories with proxy-model feedback, limiting large-scale deployment. Such rewriting may also obscure the original reasoning process and weaken transparency. Current methods therefore remain insufficient for protecting commercial LLM outputs from unauthorized distillation while preserving transparency.

To address this problem, we analyze how different components of LRM trajectories, including statement, reasoning, self-cautious, and conclusion sentences, affect inference and training on high-

[1] https://www.euaiact.com/key-issue/5

[2] https://gdpr-info.eu

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

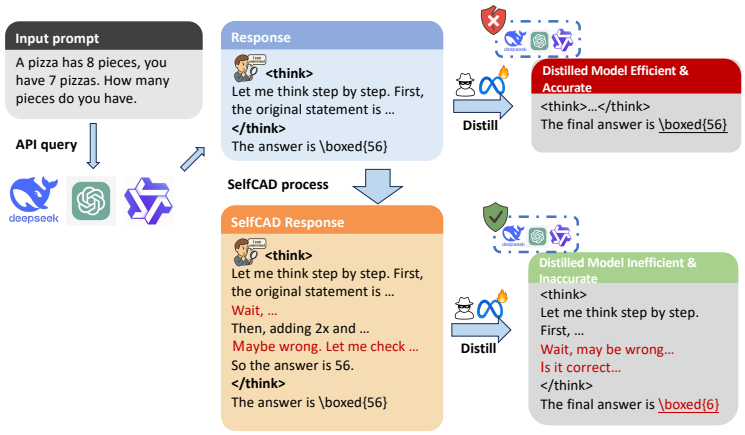


Figure 1: The overall pipeline of our SelfCAD. By inserting the model's natural self-cautious sentences into the original responses, SelfCAD can effectively degrade distilled models' performance and efficiency in a lightweight manner (can run on CPU) while preserving important reasoning trajectories.

quality reasoning data distilled by R1 (DeepSeek-AI et al., 2025). We find that the number of self-cautious sentences strongly influences both accuracy and efficiency: proper self-caution helps avoid incorrect answers, while excessive false self-caution after correct steps impairs efficiency.

Building on these observations, we propose SelfCAD (Self-Cautious Anti-Distillation), a lightweight protection shown in Fig. 1. By strategically inserting self-cautious sentences into reasoning traces, SelfCAD makes distilled models less confident even on correct steps while preserving output transparency. To improve stealthiness, these inserted sentences are collected from the protected model's own natural cautious expressions, making them better aligned with its original style. This induces redundant, inefficient reasoning, reducing accuracy and increasing output length of the distilled models while maintaining the protected models' reasoning and final results.

Contributions. Our contributions are as follows:

- We systematically analyze reasoning-trajectory components and their effects on accuracy and efficiency, showing that self-cautious sentences play a critical role in distilled models.
- We propose **SelfCAD**, a lightweight inference-time defense that proactively hinders effective output distillation and can be easily integrated into different LLM APIs.
- Experiments across various student models and distillation settings show that SelfCAD consistently increases inference cost and reduces the accuracy of unauthorized distilled models while preserving the original reasoning content for users.

Threat model. We consider a black-box API setting where a model provider releases reasoning traces to support transparency, while an unauthorized attacker collects these textual outputs to distill their student model. The attacker can observe only the protected outputs, without access to the model. The defender aims to preserve the original reasoning content and final answers for benign users while reducing the accuracy and efficiency of distilled models trained on collected outputs.

2 Related Work

2.1 Distillation Enhances Model Training

Knowledge distillation dates back to Hinton et al. (2015), which showed that transferring knowledge can improve training and performance. Later work (Goldblum et al., 2020; Gu et al., 2024; Taori et al., 2023) further demonstrated that student models benefit from teacher models, while DeepSeek-AI et al. (2025) highlighted its efficiency and potential for large-scale reasoning models.

Because training LLMs from scratch or collecting high-quality data is costly, distilling from powerful pre-trained models provides an effective shortcut. Smaller Llama (AI, 2024) and Gemma (Team et al., 2024; 2025) models, for example, heavily rely on distillation. Although this approach can reduce computation and improve performance (DeepSeek-AI et al., 2025), it raises fairness concerns for teacher-model providers and may lead to legal disputes (Times, 2025).

Distillation can use logits (Hinton et al., 2015; Goyal et al., 2025), intermediate features (Romero et al., 2015; Chen et al., 2021), or synthetic data (DeepSeek-AI et al., 2025). We focus on synthetic data because most powerful models are black-box APIs that expose only output text.

2.2 Existing Copyright Protection for LLMs

The rapid development of LLMs has intensified concerns about copyright protection (Xu et al., 2025; Jiang et al., 2024). One major line of work uses fingerprints to extract identifying signals from model parameters or behaviors, including parameter or representation analysis (Zhang et al., 2025), semantic features (Suzuki et al., 2025), and adversarial examples (Cai et al., 2025). Another line uses watermarking to embed artifacts during training (Zhang & Koushanfar, 2024; Xu et al., 2024a; Liang et al., 2024b). However, these methods often fail in black-box settings (Zhang et al., 2025), can be bypassed by fine-tuning, and mainly provide post-hoc accountability rather than proactive prevention. More directly related, Savani et al. (2025) propose anti-distillation sampling to poison reasoning traces while preserving utility, and Li et al. (2025) fine-tune the final layer with an adversarial loss against student models. Both aim to degrade distillation performance, but require modifying teacher behavior, which may introduce instability, harm performance, and weaken the original transparency of LRMs.

3 Self-Cautious Anti-Distillation

LRMs are valued for both accuracy and efficiency, but excessive generation can undermine their practical utility Si et al. (2025). Motivated by this, we analyze how components of LRM reasoning traces affect accuracy and length, and propose SelfCAD, a training-free defense that preserves reasoning transparency while hindering the performance and efficiency of the unauthorized distillation.

3.1 Reasoning Trajectories Analysis

To analyze the reasoning trajectories r generated by a reasoning model, we classify its sentences into four categories: **statement sentences** $x_s^{(i)}$, which define the problem or state a fact; **reasoning sentences** $x_r^{(i)}$, which describe computational steps or the development of a solution; **self-cautious sentences** $x_{sc}^{(i)}$, which prompt the model to check their current answer or solution; and **conclusion sentences** $x_c^{(i)}$, which provide a conclusion or the final answer. A typical reasoning trajectory starts from the statement sentence and then the other three types of sentences tend to appear in alternation at the end of reasoning, formulated as:

$$r = [x_s^{(0)}, x_r^{(0)}, x_c^{(0)}, x_{sc}^{(0)}, x_r^{(1)}, x_c^{(1)}, ...], \tag{1}$$

where $x_{s/r/sc/c}^{(i)}$ denotes i-th statement / reasoning / cautious / conclusion sentence in the reasoning trajectory. Note that the statement sentence typically appears only once at the beginning.

To evaluate the contribution of each sentence type to the end of the reasoning process, we sample a subset of reasoning traces from the public Bespoke dataset (Labs, 2025) containing high-quality reasoning outputs generated by DeepSeek-R1 (DeepSeek-AI et al., 2025), and split the traces into the defined categories. We then measure the next-token probability at the end of each sentence that belongs to the latter three sentence types with Llama3.2-1B-Instruct models (AI, 2024) on the processed dataset. This allows us to quantify how different sentence types influence the continuation of reasoning trajectories, shown in Table 1.

Table 1: The generation probabilities of ending tokens like "< |eot_id| >" following sentences of each type.

Type	Examples	Llama
Reasoning	"The left side is $(x+y)^2$..."	0.40
Self-cautious	"Wait, let's re-examine..."	0.13
Conclusion	"Then we can conclude ..."	0.86

As indicated in Table 1, most conclusion sentences terminate the reasoning trajectories. Nearly half of the reasoning sentences also serve as termination points. In contrast, self-cautious sentences exhibit a significantly lower likelihood of ending a sequence, suggesting that they are the primary driver of extended reasoning.

3.2 Training-Time Reasoning Analysis

Apart from the inference-time study, we also conduct training-time experiments to examine the impact of different reasoning components during training and distillation. Specifically, we preprocess the public Bespoke dataset (Labs, 2025) into three variants by separately removing reasoning sentences, self-cautious sentences, and conclusion sentences. The implementation details are provided in Appendix A.1. We then train Llama3.2-1B-Instruct (AI, 2024) on these datasets for 3 epochs. Finally, we evaluate their accuracy and average output length on the GSM8K benchmark with greedy sampling and zero-shot setting, listed in Table 2.

Removing self-cautious sentences yields the largest reduction in reasoning length (to 40% of the original), while removing reasoning sentences causes the largest accuracy drop. This suggests that self-cautious sentences mainly control efficiency, whereas reasoning sentences are central to accuracy. Since modifying reasoning sentences would compromise interpretability, we use self-cautious sentences as the main lever for our defense.

Table 2: Accuracy and token length for Llama3.2-1B training with reasoning data processed by modifying different components. SC denotes self-cautious.

	Acc(%)	Length
Original Data	39	3400
Remove Reasoning	33	2600
Remove Conclusion	36	4400
Remove SC	37	1300 ($\downarrow 0.4\times$)
Add SC	31	5100 ($\uparrow 1.5\times$)

We further examine the effect of inserting additional self-cautious sentences during training. As shown in Table 2, adding too many such sentences harms both performance (accuracy drops by 20%) and efficiency (generation length increases by half). This suggests that excessive self-caution, especially after correct steps, can reduce model confidence and induce unnecessary reasoning. For example, Appendix B.1 shows that even after identifying the correct answer B, the model may still excessively re-evaluating the problem.

3.3 Probabilistic Analysis for Self-Cautious Accumulation

Sections 3.1 and 3.2 show that self-cautious sentences are associated with a lower probability of termination and that adding them to distillation data produces longer student responses. We now study a simple mechanism that may contribute to this effect: repeated self-cautious continuations provide cumulative evidence in favor of further self-cautious continuation. Throughout, every continuation probability is conditioned on the complete autoregressive history, so neither stationarity nor conditional independence is assumed.

Notation and reference patterns. Let $q^{(i)}$ be the current reasoning segment, $S^{(<i)}$ all preceding segments and continuations, and $h_i = [S^{(<i)}, q^{(i)}]$ the complete history available before continuation $a^{(i)}$; $a^{(i)} \in \mathcal{A}_{\mathrm{SC}}$ means the continuation is classified as self-cautious. We introduce two reference conditional distributions, $\mathbb{P}_{\mathrm{cau}}$ (a self-cautious pattern) and $\mathbb{P}_{\mathrm{end}}$ (a terminating pattern), as analytical devices giving two reference likelihoods for the same continuation under the same history. We do not claim that an LLM contains separate internal models or switches between stationary states.

Assumption 1 (Conditional separability of self-cautious continuations). For every observed self-cautious continuation $a^{(i)} \in \mathcal{A}_{\mathrm{SC}}$, conditioned on its complete history h_i, there exists a margin $\Delta > 0$ such that

$$\log \frac{\mathbb{P}_{\mathrm{cau}}(a^{(i)} \mid h_i)}{\mathbb{P}_{\mathrm{end}}(a^{(i)} \mid h_i)} \geq \Delta. \tag{2}$$

The two probabilities may otherwise vary arbitrarily with the history; the assumption only requires a positive log-likelihood-ratio margin in favor of the cautious reference pattern. Table 1, where self-

cautious sentences have a substantially lower ending-token probability than conclusion sentences, partially supports this distinction.

Bayesian accumulation. Let $Z \in \{\text{cau}, \text{end}\}$ index the two reference patterns with prior weights $\Pr(Z = \text{cau}) = \lambda$ and $\Pr(Z = \text{end}) = 1 - \lambda$, $\lambda \in (0, 1)$. After observing k self-cautious continuations $a^{(1)}, \ldots, a^{(k)}$, each evaluated under its own complete history h_j, let

$$\pi_k = \Pr\Big(Z = \text{cau} \mid a^{(1)}, \ldots, a^{(k)}; h_1, \ldots, h_k\Big), \qquad \pi_0 = \lambda.$$

Theorem 1 (Accumulation of Self-Cautious Evidence)**.** *Under Assumption 1, the posterior weight π_k of the cautious reference pattern satisfies*

$$\pi_k \geq \frac{1}{1 + \frac{1-\lambda}{\lambda} e^{-k\Delta}}, \qquad 1 - \pi_k \leq \frac{1-\lambda}{\lambda} e^{-k\Delta}. \tag{3}$$

Thus $\pi_k \to 1$ with a finite-k error decaying exponentially in the number of observations.

Self-cautious continuation rate. For any prospective history $\widetilde{h}$, define the self-cautious continuation rates

$$\mathcal{R}_{\text{cau}}(\widetilde{h}) = \mathbb{P}_{\text{cau}}(a \in \mathcal{A}_{\text{SC}} \mid \widetilde{h}), \qquad \mathcal{R}_{\text{end}}(\widetilde{h}) = \mathbb{P}_{\text{end}}(a \in \mathcal{A}_{\text{SC}} \mid \widetilde{h}),$$

with $\mathcal{R}_{\text{cau}}(\widetilde{h}) \geq \mathcal{R}_{\text{end}}(\widetilde{h})$. The reference predictive rate after k observations is the mixture

$$\mathcal{R}_k(\widetilde{h}) = \pi_k \mathcal{R}_{\text{cau}}(\widetilde{h}) + (1 - \pi_k)\mathcal{R}_{\text{end}}(\widetilde{h}).$$

Combining this mixture with Theorem 1 gives

$$\mathcal{R}_k(\widetilde{h}) \geq \mathcal{R}_{\text{cau}}(\widetilde{h}) - \frac{1-\lambda}{\lambda} e^{-k\Delta}, \tag{4}$$

so that for any $\epsilon > 0$,

$$k \geq \frac{1}{\Delta} \log \frac{1-\lambda}{\epsilon\lambda} \quad \Longrightarrow \quad \mathcal{R}_k(\widetilde{h}) \geq \mathcal{R}_{\text{cau}}(\widetilde{h}) - \epsilon. \tag{5}$$

The full derivation is given in Appendix C.

Proof sketch. The proof is a one-line Bayes argument. In odds form, the posterior cautious odds equal the prior odds $\frac{\lambda}{1-\lambda}$ multiplied by the cumulative likelihood ratio $L_k = \prod_{j=1}^{k} \frac{\mathbb{P}_{\text{cau}}(a^{(j)}|h_j)}{\mathbb{P}_{\text{end}}(a^{(j)}|h_j)}$. Assumption 1 adds its margins in log space, $\log L_k \geq k\Delta$, hence $L_k \geq e^{k\Delta}$ and the ending weight decays as $O(e^{-k\Delta})$. Mixing the two reference rates, whose difference lies in $[0, 1]$, then yields Eq. equation 4. The full derivation is given in Appendix C.

Interpretation. Each observed self-cautious continuation contributes a positive likelihood-ratio margin; these margins accumulate multiplicatively, so the ending reference weight decays exponentially and the predictive rate moves toward $\mathcal{R}_{\text{cau}}$. Together with the lower termination probabilities in Table 1, this offers one possible explanation for why repeated self-cautious demonstrations are associated with prolonged reasoning, and motivates SelfCAD's repeated insertion strategy. The theorem describes only this sequence-level signal; it does not model how fine-tuning learns the signal or predict the magnitude of the resulting changes in length and accuracy, which are evaluated empirically below.

3.4 Our Proposed SelfCAD

Inspired by these insights, we propose **SelfCAD (Self-Cautious Anti-Distillation)**, a lightweight inference-time defense mechanism for hindering the performance and efficiency of the unauthorized distilled models. Unlike prior approaches that directly modify the behavior of distilled models (Savani et al., 2025; Li et al., 2025), our method post-processes generated outputs without affecting the teacher model's final performance. It does not rewrite or remove the original reasoning step and only inserts model-style self-cautious sentences, so the original answer path remains auditable. Since classifying the correctness of every reasoning step for each query is impractical, it is difficult

Algorithm 1 SelfCAD: Processing Reasoning Traces with Self-Cautious Sentences

Require: User query p and a pool of self-cautious sentences $\mathcal{T}_{\text{selfcautious}}$ collected from the protected model's natural outputs.

Ensure: Processed reasoning trace r_{pro} for anti-distillation while preserving transparency.

1: Generate the reasoning trace r for query p using the protected model.
2: Split r into reasoning steps: $r = [q^{(0)}, q^{(1)}, \ldots, q^{(N)}]$.
3: **for** each reasoning step $q^{(i)}$ **do**
4: Sample a sentence t_i from $\mathcal{T}_{\text{selfcautious}}$.
5: Insert t_i after $q^{(i)}$.
6: **end for**
7: Construct the processed reasoning trace: $r_{\text{pro}} = [q^{(0)}, t_0, q^{(1)}, t_1, \ldots, q^{(N)}, t_N]$.
8: **return** r_{pro}.

to accurately insert misleading self-cautious demonstrations. Instead, we adopt a simple but effective strategy: adding extra self-cautious sentences after each step, as summarized in Algorithm 1. To improve stealthiness, we collect natural self-cautious sentences from the protected model's own outputs, so the inserted sentences better match its original expression style. By exploiting the inherent shortcomings of reasoning models themselves (as discussed previously), SelfCAD provides a practical and cost-effective protection mechanism for commercial deployment. Meanwhile, it preserves the useful reasoning traces needed for users.

4 Experiments

4.1 Experiment Settings

Following standard LLM distillation, we first collect high-quality questions $\mathcal{D}_{question}$, and then prompt the teacher model to obtain its responses. Due to resource limits, we filter out incorrect or overly long responses (exceeding 4096 tokens) and obtain the final distillation dataset $\mathcal{D}_{distill}$. We then process $\mathcal{D}_{distill}$ with our SelfCAD method, as illustrated in Algorithm 1. The inserted sentence pool $\mathcal{T}$ consists of natural self-cautious expressions collected from the protected model's responses in the training set. This produces $\mathcal{D}_{distill,CAD}$ on a single CPU within several minutes.

Next, we train several student models for three epochs on $\mathcal{D}_{distill}$ and $\mathcal{D}_{distill,CAD}$ with a constant learning rate of 2×10^{-5}. All experiments are performed on NVIDIA A100 80G GPUs. We then evaluate the distilled models' reasoning accuracy and generation length to demonstrate the effectiveness of SelfCAD.

Dataset. We randomly select $30,000$ challenging questions from the BigMath dataset [3] (Albalak et al., 2025) as our question dataset $\mathcal{D}_{question}$. Big-Math is the largest open-source dataset of high-quality mathematical problems, containing over $250,000$ rigorously filtered and verified examples. After distilling responses from the teacher model and filtering out overly long or incorrect responses, the final $\mathcal{D}_{distill}$ is around $10,000$ examples.

Teacher Model. We adopt GPT-OSS-120B (OpenAI, 2025) as our teacher model, an OpenAI open-weight model with 120 billion parameters designed for strong reasoning. In our experiments, we use the medium reasoning mode because the high mode produces responses that are too long for our training and evaluation setup.

Student Model. We adopt three popular models in our main experiments. Since distillation mainly benefits smaller models, we focus on recent models from 1B to 3B parameters, including Llama-3.2-1B-Instruct, Llama-3.2-3B-Instruct (Grattafiori et al., 2024), and Qwen2.5-1.5B-Instruct (Team, 2024). These models are widely used as foundation models in both research and practical applications, making them suitable for resource-constrained scenarios. We also conduct expreiments on Qwen2.5-7B-Instruct to evaluate generalizability.

[3] https://huggingface.co/datasets/SynthLabsAI/Big-Math-RL-Verified

Evaluation Dataset. We use four widely used mathematical datasets to evaluate the distilled models' accuracy and efficiency: GSM8K (Cobbe et al., 2021), MATH-500 (Lightman et al., 2024), MATH (Hendrycks et al., 2021), and AQUA-RAT (Ling et al., 2017). We also evaluate other reasoning benchmarks in Section 5.2. All evaluations are performed in the zero-shot setting.

Baseline. We select anti-distillation sampling (Savani et al., 2025) (ADS) as a baseline. Since this method requires changing the teacher model's sampling process, we apply it only to our self-distilled dataset, using GPT-OSS-120B as the teacher model and GPT-OSS-20B as the proxy student model for distillation. Detailed experimental settings are provided in Appendix A.4.

4.2 Results on Our Distilled Setting

Table 3: Comparison of Distillation with original responses or responses generated by our SelfCAD on the BigMath dataset distilled by OSS-120B. Improvement is our gain againt original.

Benchmark	Method	Llama3.2-1B		Qwen2.5-1.5B		Llama3.2-3B	
		Accuracy	Length	Accuracy	Length	Accuracy	Length
GSM8K	Original	31%	600	55%	700	60%	400
	ADS	30%	1200	58%	1500	64%	900
	SelfCAD	23%	1300	32%	6400	38%	650
	Improvment	↓ 8%	↑ 2.2×	↓ 23%	↑ 9.1×	↓ 22%	↑ 1.6×
MATH-500	Original	15%	2300	27%	2400	28%	1900
	ADS	15%	4800	29%	5100	33%	4000
	SelfCAD	10%	4300	17%	7500	19%	4700
	Improvment	↓ 5%	↑ 2.1×	↓ 10%	↑ 3.1×	↓ 9%	↑ 2.5×
MATH	Original	31%	1300	59%	4100	54%	1000
	ADS	34%	2800	57%	2800	61%	2000
	SelfCAD	19%	3000	47%	7200	36%	2800
	Improvment	↓ 12%	↑ 2.3×	↓ 12%	↑ 1.8×	↓ 18%	↑ 2.8×
AQUA-RAT	Original	21%	1600	38%	1500	41%	1000
	ADS	19%	2300	34%	3800	39%	2500
	SelfCAD	16%	2000	27%	7000	31%	1500
	Improvment	↓ 5%	↑ 1.3×	↓ 11%	↑ 4.7×	↓ 10%	↑ 1.5×
Avg Improvement		↓ 7.5%	↑ 2.0×	↓ 14%	↑ 4.7×	↓ 14.8%	↑ 2.1×

We fine-tune three student models on both the original distilled dataset and the dataset processed by SelfCAD, and then evaluate them on four benchmark datasets. The experimental results are presented in Table 3. Most student models exhibit a clear accuracy drop of approximately 10 percentage points. Moreover, the generated output lengths increase substantially, ranging from $1.3\times$ to $9.1\times$ those of normally distilled models. Notably, after SelfCAD processing, the average response length of Qwen2.5-1.5B-Instruct on GSM8K increases from 700 tokens to 6400 tokens, while its accuracy decreases from 55% to 32%. This substantial increase in length, together with the decrease in accuracy, produces inefficient and less useful reasoning traces for distilled student models, degrading the performance and efficiency unauthorized distilled models. For the baseline method ADS, the performance is unstable: it yields only marginal accuracy decreases on some benchmarks and even improves performance in some cases. We find that this failure occurs because long reasoning traces can eventually recover the correct answer in ADS, leading to correct final results. Such complex reasoning behavior is not evaluated in their work.

4.3 Results on Public Distilled Dataset

In addition to our replicated distillation process, we conduct experiments on Bespoke Labs (2025), a public high-quality reasoning dataset on HuggingFace distilled from R1. We apply the same SelfCAD procedure to this dataset and train models on both the original Bespoke dataset and its SelfCAD-processed counterpart, following the setting described above. The results are listed in Table 4.

Responses generated by R1 are generally longer than those produced by GPT-OSS-120B, which in turn leads to longer outputs from student models distilled on the vanilla Bespoke dataset. For example, after fine-tuning Qwen2.5-1.5B-Instruct on our own distilled dataset, the average response

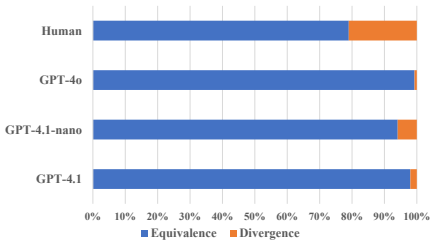
Table 4: Comparison of distillation with original responses or responses generated by SelfCAD on the Bespoke dataset distilled by R1.

Benchmark	Method	Llama3.2-1B		Qwen2.5-1.5B		Llama3.2-3B	
		Accuracy	Length	Accuracy	Length	Accuracy	Length
GSM8K	Original	34%	3400	66%	3000	73%	2000
	SelfCAD	21%	4800	58%	6000	61%	2600
	Improvment	↓ 13%	↑ 1.4×	↓ 8%	↑ 2.0×	↓ 12%	↑ 1.3×
MATH-500	Original	33%	4000	19%	6000	38%	4500
	SelfCAD	28%	6000	11%	9000	31%	8000
	Improvment	↓ 5%	↑ 1.5×	↓ 8%	↑ 1.5×	↓ 7%	↑ 1.8×
MATH	Original	35%	6000	63%	4000	69%	4500
	SelfCAD	31%	8000	57%	5500	59%	6000
	Improvment	↓ 4%	↑ 1.3×	↓ 6%	↑ 1.4×	↓ 10%	↑ 1.3×
AQUA-RAT	Original	18%	6000	39%	3800	42%	5000
	SelfCAD	9%	8200	32%	5500	36%	6600
	Improvment	↓ 9%	↑ 1.4×	↓ 7%	↑ 1.5×	↓ 6%	↑ 1.3×
Avg Improvement		↓ 7.8%	↑ 1.4×	↓ 6.8%	↑ 1.6×	↓ 8.8%	↑ 1.4×

Table 5: Comparison of distillation with original responses or responses generated by SelfCAD on the Bespoke dataset with Qwen2.5-7B-Instruct.

Benchmark	Original		SelfCAD	
	Accuracy	Length	Accuracy	Length
GSM8K	92%	1400	83%	2300
MATH-500	70%	3000	61%	3900
MATH	78%	2000	69%	2800
AQUA-RAT	61%	2700	52%	4300

Figure 2: Semantic consistency of reasoning traces before and after processing with SelfCAD.



length on the MATH-500 evaluation dataset is 2,400 tokens. In contrast, when fine-tuned on the Bespoke dataset, the average response length on MATH-500 increases to 6,000 tokens. As a result, applying SelfCAD to the Bespoke dataset yields a less pronounced increase in response length than applying it to our own distilled dataset. This is partly because we cap generated responses at 10,000 tokens. Thus, some longer responses are truncated, limiting the observed effect of SelfCAD. Nevertheless, the average response lengths still increase to 1.4–1.6× their original lengths, while accuracy drops around 7 percentage points across evaluation datasets. A possible explanation for this accuracy drop is that excessive self-cautious behavior weakens model confidence and increases the likelihood of incorrect conclusions.

4.4 Results on Larger Models

To demonstrate the generalizability of SelfCAD, we conduct the same experiment using the Qwen2.5-7B-Instruct model on the Bespoke dataset. Results presented in Table 5 show a similar effect, with a 10 percentage point accuracy drop and around a 1.5× length increase, demonstrating that our method generalizes to larger models.

5 Ablation Studies

5.1 Ablation Study on Method Stealthiness

LLM evaluation. To evaluate stealthiness, i.e., whether SelfCAD remains undetected during distillation, we use GPT-4o, GPT-4.1-nano, and GPT-4.1 to identify potential semantic discrepancies before and after processing. The detection prompt is provided in Appendix A.3. As shown in Fig. 2, SelfCAD preserves strong semantic integrity, achieving a 99.3% equivalence rate judged by GPT-

Table 6: Accuracy and response length on four non-math datasets with Qwen2.5-1.5B-Instruct.

Method	Metric	Folio	PiQA	ProofWriter	SiQA
Original	Acc	33%	53%	47%	62%
	Length	3500	3300	3200	1200
SelfCAD	Acc	15%	40%	31%	50%
	Length	8000	6000	7200	3000
Imp	Δ_{Acc}	-18%	13%	16%	12%
	Δ_{Len}	2.3×	1.8×	2.3×	2.5×

Table 7: Response lengths and accuracies on GSM8K with Qwen2.5-1.5B-Instruct under adaptive strategies.

	Length	Accuracy
Original	700	55
SelfCAD	6400	32 (↓ 23%)
SelfCAD + Efficient Prompt	5500	31 (↓ 24%)
SelfCAD + DEER	3200	29 (↓ 26%)
SelfCAD + Rewrite	500	15 (↓ 40%)
SelfCAD + Remove	20	12 (↓ 43%)

4o. Results show that SelfCAD preserves the semantic information in original reasoning traces and difficult for state-of-the-art LLMs to detect.

Human evaluation. We sample 100 data points and ask three researchers to annotate each by majority vote using three options: A. affects readability, B. does not affect readability but differences are noticeable, and C. almost no difference. Option A accounts for 0%, showing that SelfCAD does not affect normal comprehension; Options B and C account for 21% and 79%, respectively, suggesting that our approach remains largely unnoticed.

5.2 Ablation Study on Datasets beyond Math

To demonstrate that our method can generalize to more domains, we conducted evaluations using the Qwen2.5-1.5B model on four datasets: Folio, PIQA, Proof Writer, and SocialQA. The experimental results are shown in Table 6. From the results, it can be observed that our SelfCAD consistently renders the distilled model ineffective.

5.3 Ablation Study on Adaptive Distillation

We further examine whether attackers can bypass SelfCAD using several adaptive strategies.

Efficient reasoning after distillation. We first evaluate whether efficiency reasoning methods can mitigate SelfCAD. Specifically, we apply DEER (Yang et al., 2025), an inference-time truncation method for removing unnecessary thoughts, and also prompt the distilled model with "Please reason efficiently and avoid redundant checking" (denoted as Efficient Prompt). As shown in Table 7, neither strategy bypasses SelfCAD: although both reduce response length compared with SelfCAD, the outputs remain much longer than original baseline and accuracy remains substantially degraded.

Rewriting or removing CoT for distillation. We also consider preprocessing-based attacks. Rewriting protected responses is a natural bypass strategy, but it is resource-intensive and depends heavily on the rewriting model. In addition, we evaluate a more aggressive setting that removes all poisoned reasoning traces before distillation. As shown in Table 7, both strategies substantially reduce accuracy, producing student models that are much less useful than those trained on the original data. Thus, rewriting or removing poisoned CoT does not effectively bypass SelfCAD in our setting. The reason is that these preprocessing strategies substantially alter or discard the protected model's original reasoning traces, which are precisely the knowledge that the attacker aims to distill.

6 Conclusions

In this work, we systematically analyze reasoning trajectories and identify the crucial role of the number of self-cautious sentences in determining the efficiency and accuracy of distilled models. Building on this insight, we propose SelfCAD, a lightweight inference-time defense against unauthorized distillation that can be directly applied to existing LRMs. Extensive experiments confirm that SelfCAD effectively degrades unauthorized distilled models by increasing inference costs and reducing accuracy, while maintaining original reasoning steps for end users. Our results highlight a practical path toward protecting reasoning models and encouraging transparency in leading LLMs without the risk of misuse through distillation. We believe SelfCAD can inspire future research in trustworthy and responsible AI deployment, fostering a more secure ecosystem for intellectual property protection.

AI USE STATEMENT

In this work, we used generative AI tools to assist with data annotation. All AI-generated annotations were reviewed and verified by the authors before being used in our experiments. We did not use generative AI tools to formulate mathematical claims, develop proofs, propose or refine hypotheses, design the research methodology or experiments, or interpret the experimental results; the remaining required-disclosure tasks are not applicable to this work. Additionally, we used generative AI tools to assist with writing and editing software code and to improve the readability and language of the manuscript. All AI-assisted code was inspected and tested by the authors, and all AI-edited text was carefully reviewed to ensure technical accuracy and fidelity to the authors' intended meaning. The authors take full responsibility for the final content of this work, including all text, claims, code, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work studies defenses against unauthorized distillation of reasoning-model outputs. The intended use of SelfCAD is to help model providers protect intellectual property while continuing to expose reasoning traces for transparency. However, output manipulation methods may also affect user experience or be misused to intentionally reduce the utility of downstream training data. We therefore emphasize that SelfCAD should be deployed only by authorized model providers on their own outputs, with appropriate disclosure and careful calibration of protection strength. Our experiments use publicly available datasets and models, and no private or sensitive information is involved.

REPRODUCIBILITY STATEMENT

We use publicly available datasets and open-weight models throughout our experiments. Section 4.1 describes the datasets, teacher and student models, evaluation protocol, and primary training configuration. Appendix A provides additional implementation details, including the reasoning-trace classification and data-construction procedure, prompts, hyperparameters, and baseline settings. We report results across multiple model families, model scales, datasets, and training configurations to facilitate verification of our findings. The assumptions and full proof of our theoretical result are provided in Appendix C. Upon publication, we will release the source code, data-processing scripts, experiment configurations, and instructions needed to reproduce the reported results.

REFERENCES

Meta AI. Llama 3.2: Revolutionizing edge ai and vision with open, customizable models, 2024. URL https://ai.meta.com/blog/llama-3-2-connect-2024-vision-edge-mobile-devices/.

Alon Albalak, Duy Phung, Nathan Lile, Rafael Rafailov, Kanishk Gandhi, Louis Castricato, Anikait Singh, Chase Blagden, Violet Xiang, Dakota Mahan, and Nick Haber. Big-math: A large-scale, high-quality math dataset for reinforcement learning in language models, 2025.

Jiacheng Cai, Jiahao Yu, Yangguang Shao, Yuhang Wu, and Xinyu Xing. Utf: Under-trained tokens as fingerprints——a novel approach to llm identification. In *Proceedings of the The First Workshop on LLM Security (LLMSEC)*, pp. 1–6, 2025.

Pengguang Chen, Shu Liu, Hengshuang Zhao, and Jiaya Jia. Distilling knowledge via knowledge review. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 5008–5017, 2021.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Ziyao Cui, Minxing Zhang, and Jian Pei. On membership inference attacks in knowledge distillation. *arXiv preprint arXiv:2505.11837*, 2025.

Sumit Kumar Dam, Choong Seon Hong, Yu Qiao, and Chaoning Zhang. A complete survey on llm-based ai chatbots. *arXiv preprint arXiv:2406.16937*, 2024.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, and et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025.

Micah Goldblum, Liam Fowl, Soheil Feizi, and Tom Goldstein. Adversarially robust distillation. In *AAAI*, 2020.

Sachin Goyal, David Lopez-Paz, and Kartik Ahuja. Distilled pretraining: A modern lens of data, in-context learning and test-time scaling. *arXiv preprint arXiv:2509.01649*, 2025.

Aaron Grattafiori, Abhimanyu Dubey, and et al. The llama 3 herd of models, 2024.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. Minillm: Knowledge distillation of large language models. In *ICLR*, 2024.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. In *NeurIPS*, 2021.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015.

Yongqi Jiang, Yansong Gao, Chunyi Zhou, Hongsheng Hu, Anmin Fu, and Willy Susilo. Intellectual property protection for deep learning model and dataset intelligence, 2024.

Bespoke Labs. Bespoke-stratos: The unreasonable effectiveness of reasoning distillation. https://www.bespokelabs.ai/blog/bespoke-stratos-the-unreasonable-effectiveness-of-reasoning-distillation, 2025. Accessed: 2026-09-26.

Pingzhi Li, Zhen Tan, Huaizhi Qu, Huan Liu, and Tianlong Chen. Doge: Defensive output generation for llm protection against knowledge distillation, 2025.

Jiacheng Liang, Ren Pang, Changjiang Li, and Ting Wang. Model extraction attacks revisited. In *Proceedings of the 19th ACM Asia Conference on Computer and Communications Security*, pp. 1231–1245, 2024a.

Yuqing Liang, Jiancheng Xiao, Wensheng Gan, and Philip S. Yu. Watermarking techniques for large language models: A survey, 2024b.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, volume 2024, pp. 39578–39601, 2024.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Proceedings of the 55th annual meeting of the association for computational linguistics (volume 1: Long papers)*, pp. 158–167, 2017.

Daye Nam, Andrew Macvean, Vincent Hellendoorn, Bogdan Vasilescu, and Brad Myers. Using an llm to help with code understanding. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, pp. 1–13, 2024.

OpenAI. GPT-4 Technical Report. *CoRR abs/2303.08774*, 2023.

OpenAI. gpt-oss-120b & gpt-oss-20b model card, 2025.

Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets, 2015. URL `https://arxiv.org/abs/1412.6550`.

Yash Savani, Asher Trockman, Zhili Feng, Yixuan Even Xu, Avi Schwarzschild, Alexander Robey, Marc Finzi, and J. Zico Kolter. Antidistillation sampling, 2025.

Wai Man Si, Mingjie Li, Michael Backes, and Yang Zhang. Excessive reasoning attack on reasoning llms. *arXiv preprint arXiv:2506.14374*, 2025.

Teppei Suzuki, Ryokan Ri, and Sho Takase. Natural fingerprints of large language models. *arXiv preprint arXiv:2504.14871*, 2025.

Mark Sweney and Dan Milmo. Openai 'reviewing' allegations that its ai models were used to make deepseek. https://www.theguardian.com/technology/2025/jan/29/openai-chatgpt-deepseek-china-us-ai-models, 2025.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, and Percy Liang. Stanford alpaca: An instruction-following LLaMA model. *arXiv preprint arXiv:2304.08177*, 2023.

Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, and et al. Gemma: Open models based on gemini research and technology, 2024.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, Louis Rouillard, and et al. Gemma 3 technical report, 2025.

Qwen Team. Qwen2.5: A party of foundation models, September 2024. URL https://qwenlm.github.io/blog/qwen2.5/.

New York Times. Openai says it has evidence china's deepseek used its model to train competitor, 2025. URL https://www.nytimes.com/2025/01/29/technology/openai-deepseek-data-harvest.html.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. LLaMA: Open and Efficient Foundation Language Models. *CoRR abs/2302.13971*, 2023.

Florian Tramèr, Fan Zhang, Ari Juels, Michael K Reiter, and Thomas Ristenpart. Stealing machine learning models via prediction {APIs}. In *25th USENIX security symposium (USENIX Security 16)*, pp. 601–618, 2016.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-instruct: Aligning language models with self-generated instructions. *arXiv preprint arXiv:2212.10560*, 2022.

Jiashu Xu, Fei Wang, Mingyu Derek Ma, Pang Wei Koh, Chaowei Xiao, and Muhao Chen. Instructional fingerprinting of large language models. In *NAACL*, 2024a.

Xiaohan Xu, Ming Li, Chongyang Tao, Tao Shen, Reynold Cheng, Jinyang Li, Can Xu, Dacheng Tao, and Tianyi Zhou. A survey on knowledge distillation of large language models. *arXiv preprint arXiv:2402.13116*, 2024b.

Zhenhua Xu, Xubin Yue, Zhebo Wang, Qichen Liu, Xixiang Zhao, Jingxuan Zhang, Wenjun Zeng, Wengpeng Xing, Dezhang Kong, Changting Lin, and Meng Han. Copyright protection for large language models: A survey of methods, challenges, and trends, 2025.

Chenxu Yang, Qingyi Si, Yongjie Duan, Zheliang Zhu, Chenyu Zhu, Zheng Lin, Li Cao, and Weiping Wang. Dynamic early exit in reasoning models. *arXiv preprint arXiv:2504.15895*, 2025.

Jie Zhang, Dongrui Liu, Chen Qian, Linfeng Zhang, Yong Liu, Yu Qiao, and Jing Shao. REEF: representation encoding fingerprints for large language models. In *ICLR*, 2025.

Ruisi Zhang and Farinaz Koushanfar. Emmark: Robust watermarks for ip protection of embedded quantized large language models. In *Proceedings of the 61st ACM/IEEE Design Automation Conference*, pp. 1–6, 2024.

A Implementation Details

A.1 Classification Methodology for Reasoning Traces

Classification Prompt Template

You are an AI assistant specialized in text analysis and structured output generation. Your task is to analyze a given prompt-response pair and categorize each part of the response into specific types based on its function. Use the following fixed categories:

1. Averaging Statement: Sentences that restate or summarize information from the prompt or previous context.
2. Reasoning/Planning: Parts that involve logical steps, equations, calculations, or plans to approach the problem.
3. Self-Checking/Self-Cautious: Statements where the model verifies its own reasoning, expresses uncertainty, or double-checks assumptions.
4. Result Statement: The final part where the answer or conclusion is clearly stated (e.g., starting with "The answer is" or similar).

Output Format:
Use a numbered list for each categorized part, strictly in the order they appear in the response. For each part, specify:
The exact text snippet from the response.
Its category label (from the list above).
If a part spans multiple sentences but belongs to the same category, group them together under one entry.
Do not add any extra commentary or analysis beyond the categorization.

Example:
Prompt: "Calculate the area of a circle with radius 5."
Response: "To find the area of a circle, we use the formula A = πr^2. Here, the radius is 5. So, A = $\pi 5^2$ = 25π. Let me verify: π is approximately 3.14, so 25 * 3.14 $\approx$ 78.5. The answer is 25π or approximately 78.5 units^2."

Categorized Output:
1. [To find the area of a circle, we use the formula A = πr^2.] - Reasoning/Planning
2. [Here, the radius is 5.] - Averaging Statement
3. [So, A =$\pi 5^2$ = 25π.] - Reasoning/Planning
4. [Let me verify: π is approximately 3.14, so 25 * 3.14 $\approx$ 78.5.] - Self-Checking/Self-Cautious
5. [The answer is 25π or approximately 78.5 units^2.] - Result Statement

Now, analyze the following prompt-response pair and provide the categorized output as described.

Prompt:

{prompt}

Response:

{response}

We use GPT-4.1 to partition reasoning traces into several segments and assign a category label to each segment, with the user prompt template as above. Then, we use pattern matching to extract all sentences and their corresponding categories. After that, we construct four datasets as follows:

- A dataset where Reasoning Sentences are removed and the remaining sentences are concatenated in order.

- A dataset where Self-cautious Sentences are removed and the remaining sentences are concatenated in order.
- A dataset where Conclusion Sentences are removed and the remaining sentences are concatenated in order.
- A dataset where all sentences are retained and concatenated in order.

Subsequently, we fine-tune the student model using these four datasets for experimentation.

A.2 Hyperparameters and Experimental Settings

For all teacher-model generations, we use vLLM with the sampling temperature set to 0.7. For student-model training, we use a batch size of 64. We try learning rates of 1×10^{-5} and 2×10^{-5}, and find that the experimental results are robust across these settings. All experiments are conducted on NVIDIA A100 GPUs.

A.3 Template of Judge Model

Judgment Prompt Template

I will provide you with a query and two responses. Your task is to determine whether these two responses have any semantic differences, disregarding any differences in form or expression. Please reply only with "yes" if there is a semantic difference or "no" if there is no semantic difference. Do not include any additional content in your response.

Query:
{query}

Response 1:
{response1}

Response 2:
{response2}

Now please determine whether these two responses have any semantic differences. Only reply with yes or no.

To guarantee the fairness of the judgment, the original and processed responses are formatted into the template above and presented in a randomized order. We then extract the "yes" or "no" from the API's response.

A.4 Baseline Selection and Implementation

We consider two existing anti-distillation strategies that are closely related to our setting: DOGE (Li et al., 2025) and anti-distillation sampling (Savani et al., 2025). For DOGE, we found it difficult to reproduce the reported results reliably because its optimization is sensitive to hyperparameters and the training process is unstable in our setting. Therefore, we use anti-distillation sampling as our main baseline.

However, we further observe that anti-distillation sampling is also sensitive to both hyperparameters and model choices. Following the original paper, we set $\tau = 0.6$ and $\epsilon = 0.01$, and vary λ to control the strength of anti-distillation sampling. Table 8 reports the teacher-model accuracy on BigMath under different values of λ. The results show that the teacher-model accuracy changes sensitively with λ, indicating that this baseline requires careful tuning to preserve the utility of the teacher model.

In our experiments, we finally choose $\lambda = 0.012$ as the baseline setting. Although the teacher-model accuracy remains high when $\lambda = 0.015$, we find that the think tokens in about 90% of the generated data cannot be parsed correctly, which prevents us from reliably extracting the chain-of-thought. We therefore adopt the slightly weaker setting $\lambda = 0.012$, which better balances anti-distillation strength with parseability of the generated reasoning traces.

Table 8: Teacher-model accuracy on BigMath under different anti-distillation sampling strengths. We follow the original setting with $\tau = 0.6$ and $\epsilon = 0.01$, and vary λ.

λ	0.0	0.01	0.012	0.015	0.018	0.02
Acc.	60.7	60.7	59.2	53.6	29.2	16.9

B Additional Results

B.1 Full Content of Over-Cautious Example

Full Content of over-cautious example

User: Return your final response within \boxed{}. In a certain year the price of gasoline rose by 20% during January, fell by 20% during February, rose by 25% during March, and fell by x% during April. The price of gasoline at the end of April was the same as it had been at the beginning of January. To the nearest integer, what is x
(A) 12 (B) 17 (C) 20 (D) 25 (E) 35

Assistant: Okay, so I need to figure out the value of x, the percentage that gasoline prices fell in April, such that the price at the end of April is the same as it was at the beginning of January. The problem gives me a series of percentage changes each month: a 20% increase in January, a 20% decrease in February, a 25% increase in March, and then an x% decrease in April. The answer choices are given, so I need to work through the problem step by step.

Full Content of over-cautious example

First, let me recall that percentage changes are multiplicative. So if something increases by 20%, that's multiplying by 1.20, and if it decreases by 20%, that's multiplying by 0.80. Similarly, a 25% increase is multiplying by 1.25, and a decrease of x% would be multiplying by (1 - x/100). So the overall effect of these four changes should result in no net change, meaning the product of these multipliers is 1. Let me denote the initial price as P. Then:
- After January's 20% increase, the price is P * 1.20.
- After February's 20% decrease, it becomes P * 1.20 * 0.80.
- After March's 25% increase, it's P * 1.20 * 0.80 * 1.25.
- After April's x% decrease, the final price is P * 1.20 * 0.80 * 1.25 * (1 - x/100).
Since the final price equals the initial price P, we have:
1.20 * 0.80 * 1.25 * (1 - x/100) = 1
So I need to compute 1.20 * 0.80 * 1.25 first and then solve for x. Let me compute this step by step. First, 1.20 * 0.80. Let's do that: 1.20 * 0.80 = (1 + 0.20) * 0.80 = 1 * 0.80 + 0.20 * 0.80 = 0.80 + 0.16 = 0.96. So after January and February, the price is 96% of the original. Then, a 25% increase in March: 0.96 * 1.25. Let's compute that.
0.96 * 1.25. Hmm, 1.25 is 5/4, so 0.96 * 5/4 = (0.96 / 4) * 5 = 0.24 * 5 = 1.20
Wait, that's interesting. So after March, the price is 1.20 times the original price? But wait, hold on. Let me check that again.
Wait, 0.96 * 1.25. Maybe another way: 0.96 * 1 = 0.96, 0.96 * 0.25 = 0.24, so total is 0.96 + 0.24 = 1.20. Yeah, that's correct. So after three months, the price is 1.20P. Then in April, it falls by x%, resulting in P. So the multiplier for April is (1 - x/100), and 1.20 * (1 - x/100) = 1.
Therefore, solving for (1 - x/100) = 1 / 1.20 = 5/6 ≈ 0.8333. So 1 - x/100 ≈ 0.8333, so x/100 ≈ 1 - 0.8333 ≈ 0.1666. Therefore, x ≈ 16.66%. The nearest integer is 17%, so the answer is B.
Wait, but let me check again step by step to make sure I didn't make any mistakes.
Starting with P: January: +20% means P * 1.20. February: -20% means 1.20P * 0.80 = 0.96P. March: +25% means 0.96P * 1.25 = 1.20P. April: -x% means 1.20P * (1 - x/100) = P. So, 1.20 * (1 - x/100) = 1. Therefore, (1 - x/100) = 1/1.20 = 5/6 ≈ 0.833333... So, x/100

= 1 - 5/6 = 1/6 ≈ 0.166666... Multiply by 100: x ≈ 16.666..., which rounds to 17%. So the answer is B.
But let me verify with an example. Let's assume the initial price is $100. January: 20% increase: 100 + 20 = $120. February: 20% decrease: 120 - 24 = $96. March: 25% increase: 96 + 24 = $120. April: Need to decrease from $120 to $100. What percentage is that? The decrease is $20 from $120. So 20/120 = 1/6 ≈ 0.166666..., which is 16.666...%, so 17% when rounded to the nearest integer. That's consistent with the previous calculation. Therefore, the answer is B) 17.
But wait, the answer choices are given, and 17 is option B, which matches. So I think that's correct. However, just to make sure, let's check if perhaps there was a miscalculation in the multipliers. Alternatively, maybe the problem is presented in a different way. Let me check the problem statement again: Ïn a certain year the price of gasoline rose by 20% during January, fell by 20% during February, rose by 25% during March, and fell by x% during April. The price of gasoline at the end of April was the same as it had been at the beginning of January. To the nearest integer, what is x?Ÿes, that's exactly the scenario. So starting price P, after Jan +20%, Feb -20%, March +25%, April -x%, ending at P. Therefore, the cumulative effect is as we calculated. So, the calculation seems correct. The steps are: 1.20 * 0.80 = 0.96, 0.96 * 1.25 = 1.20, 1.20 * (1 - x/100) = 1. Therefore, (1 - x/100) = 1 / 1.20 ≈ 0.8333, x ≈ 16.666..., which rounds to 17. Therefore, answer is B. Alternatively, maybe someone might think that the percentages are compounded differently, but the standard way is to apply each percentage successively to the current price. So 20% increase, then 20% decrease of the new price, etc. So the calculations should be correct.
Another way to think about it is: After January: 1.2. After February: 1.2 * 0.8 = 0.96. After March: 0.96 * 1.25 = 1.2. After April: 1.2 * (1 - x/100) = 1. Therefore, same result. Thus, x ≈ 16.666...%, so 17%. Yes, that seems correct. So I'm confident the answer is B. **Final Answer** \boxed{B}

B.2 Experiment Results for Each Epoch

Table 9: Experiment results for each epoch with Qwen2.5-1.5B

Options	Original (3epoch)	SelfCAD (1epoch)	SelfCAD (2epoch)	SelfCAD (3epoch)	SelfCAD (4epoch)
Response Length	600	2100	2600	2900	3200

To demonstrate the effectiveness of SelfCAD throughout the training process, we evaluate the model after each epoch of training, with the response length shown in Table 9. The results indicate that our method consistently increases the model's output length.

C Proof of Theorem 1

We collect here the full derivation underlying Theorem 1 and the rate bound Eq. equation 4.

Step 1: Sequential likelihood ratio. For k observed self-cautious continuations $a^{(1)}, \dots, a^{(k)}$, each evaluated under its complete autoregressive history h_j, define the cumulative conditional likelihood ratio

$$L_k = \prod_{j=1}^{k} \frac{\mathbb{P}_{\text{cau}}(a^{(j)} \mid h_j)}{\mathbb{P}_{\text{end}}(a^{(j)} \mid h_j)}. \tag{6}$$

This is a sequential likelihood ratio, *not* an independence factorization: every term is conditioned on the full history h_j. Applying Assumption 1 term by term,

$$\begin{aligned} \log L_k &= \sum_{j=1}^{k} \log \frac{\mathbb{P}_{\text{cau}}(a^{(j)} \mid h_j)}{\mathbb{P}_{\text{end}}(a^{(j)} \mid h_j)} \\ &\geq k\Delta, \end{aligned} \tag{7}$$

and therefore

$$L_k \geq e^{k\Delta}. \tag{8}$$

Step 2: Posterior cautious weight. With prior weights $\Pr(Z = \text{cau}) = \lambda$ and $\Pr(Z = \text{end}) = 1 - \lambda$, Bayes' rule in odds form gives

$$\frac{\pi_k}{1 - \pi_k} = \frac{\lambda}{1 - \lambda} L_k, \tag{9}$$

so that

$$\pi_k = \frac{\lambda L_k}{\lambda L_k + (1 - \lambda)}. \tag{10}$$

Using $L_k \geq e^{k\Delta}$ from Eq. equation 8,

$$\begin{aligned} \pi_k &= \frac{1}{1 + \frac{1-\lambda}{\lambda} L_k^{-1}} \\ &\geq \frac{1}{1 + \frac{1-\lambda}{\lambda} e^{-k\Delta}}, \end{aligned} \tag{11}$$

which proves the first inequality of Theorem 1. Moreover,

$$1 - \pi_k = \frac{1 - \lambda}{\lambda L_k + (1 - \lambda)} \leq \frac{1 - \lambda}{\lambda} e^{-k\Delta}, \tag{12}$$

which proves the second inequality.

Step 3: Reference predictive rate. For any prospective history $\widetilde{h}$, the two reference rates satisfy

$$\mathcal{R}_{\text{cau}}(\widetilde{h}) = \mathbb{P}_{\text{cau}}(a \in \mathcal{A}_{\text{SC}} \mid \widetilde{h}), \qquad \mathcal{R}_{\text{end}}(\widetilde{h}) = \mathbb{P}_{\text{end}}(a \in \mathcal{A}_{\text{SC}} \mid \widetilde{h}),$$

and by assumption $\mathcal{R}_{\text{cau}}(\widetilde{h}) \geq \mathcal{R}_{\text{end}}(\widetilde{h}) \geq 0$. The predictive rate after k observations is the mixture

$$\mathcal{R}_k(\widetilde{h}) = \pi_k \mathcal{R}_{\text{cau}}(\widetilde{h}) + (1 - \pi_k)\mathcal{R}_{\text{end}}(\widetilde{h}). \tag{13}$$

Rearranging,

$$\mathcal{R}_{\text{cau}}(\widetilde{h}) - \mathcal{R}_k(\widetilde{h}) = (1 - \pi_k)\big[\mathcal{R}_{\text{cau}}(\widetilde{h}) - \mathcal{R}_{\text{end}}(\widetilde{h})\big]. \tag{14}$$

Because $0 \leq \mathcal{R}_{\text{end}}(\widetilde{h}) \leq \mathcal{R}_{\text{cau}}(\widetilde{h}) \leq 1$, the bracketed difference lies in $[0, 1]$. Combining this with Eq. equation 12 yields

$$\mathcal{R}_k(\widetilde{h}) \geq \mathcal{R}_{\text{cau}}(\widetilde{h}) - \frac{1 - \lambda}{\lambda} e^{-k\Delta}, \tag{15}$$

which proves Eq. equation 4. Finally, if

$$k \geq \frac{1}{\Delta} \log \frac{1 - \lambda}{\epsilon\lambda}, \tag{16}$$

then $\frac{1-\lambda}{\lambda} e^{-k\Delta} \leq \epsilon$, and hence $\mathcal{R}_k(\widetilde{h}) \geq \mathcal{R}_{\text{cau}}(\widetilde{h}) - \epsilon$.