Self-Training Large Language Models with Confident Reasoning

Hyosoon Jang[1], **Yunhui Jang**[2], **Sungjae Lee**[1], **Jungseul Ok**[1], **Sungsoo Ahn**[2]
[1]POSTECH [2]KAIST
{hsjang1205,sungjaelee25,jungseul}@postech.ac.kr,
{yunhuijang,sungsoo.ahn}@kaist.ac.kr

Abstract

Large language models (LLMs) have shown impressive performance by generating reasoning paths before final answers, but learning such a reasoning path requires costly human supervision. To address this issue, recent studies have explored self-training methods that improve reasoning capabilities using pseudo-labels generated by the LLMs themselves. Among these, confidence-based self-training fine-tunes LLMs to prefer reasoning paths with high-confidence answers, where confidence is estimated via majority voting. However, such methods exclusively focus on the quality of the final answer and may ignore the quality of the reasoning paths, as even an incorrect reasoning path leads to a correct answer by chance. Instead, we advocate the use of reasoning-level confidence to identify high-quality reasoning paths for self-training, supported by our empirical observations. We then propose a new self-training method, **CORE-PO**, that fine-tunes LLMs to prefer high-**CO**nfidence **RE**asoning paths through **P**olicy **O**ptimization. Our experiments show that CORE-PO improves the accuracy of outputs on four in-distribution and two out-of-distribution benchmarks, compared to existing self-training methods.

1 Introduction

Large language models (LLMs) have shown impressive performance across various tasks by generating reasoning paths before yielding the final answer (Wei et al., 2022; Kojima et al., 2022; Zhang et al., 2023). However, the potential for improving reasoning through supervision is limited by the scarcity of high-quality data with human-annotated or ground-truth labels. To address this issue, recent studies have proposed *self-training methods* for LLMs, which leverage pseudo-labels generated by the LLMs themselves and require only the input questions (Huang et al., 2023; Kumar et al., 2024; Prasad et al., 2024; Zhang et al., 2024c,b; Ranaldi and Freitas, 2024; Zuo et al., 2025).[1] At a high level, these methods fine-tune the base LLMs to prefer high-quality outputs identified through self-assessment strategies in inference-time scaling techniques, e.g., self-consistency (Wang et al., 2023), tree-of-thoughts (Yao et al., 2023), and self-refinement (Madaan et al., 2023a).

As a representative approach, confidence-based methods have improved the reasoning capabilities by training LLMs to prefer reasoning paths associated with high-confidence answers (Huang et al., 2023; Prasad et al., 2024; Zhang et al., 2024b; Zuo et al., 2025). These methods are motivated by the observation that the answers with high confidence scores tend to yield high accuracy (Wang et al., 2023; Taubenfeld et al., 2025), and thus assume that reasoning paths leading to such answers are reliable. Specifically, Huang et al. (2023); Prasad et al. (2024) and Zhang et al. (2024b) estimate the confidence in answers using self-consistency scores measured via majority voting, and then fine-tune LLMs to prefer reasoning paths associated with high-confidence answers.

In this work, we argue that existing confidence-based self-training methods exclusively focus on *answer-level confidence* and may ignore the quality of the reasoning path. In practice, as illustrated in Figure 1 and observed in Observation 1, LLMs often generate incorrect reasoning paths that lead to high-confidence answers, even when those answers are correct (Lanham et al., 2023; Zhang et al., 2024a). Consequently, LLMs may learn to prefer incorrect reasoning paths associated with high-confidence answers, which degrade their reasoning capabilities. This pitfall highlights the necessity of incorporating reasoning-aware confidence measures into the self-training of LLMs to better identify high-quality reasoning paths.

[1]We refer to self-training as a scheme that requires only the input questions, without labels or external models.

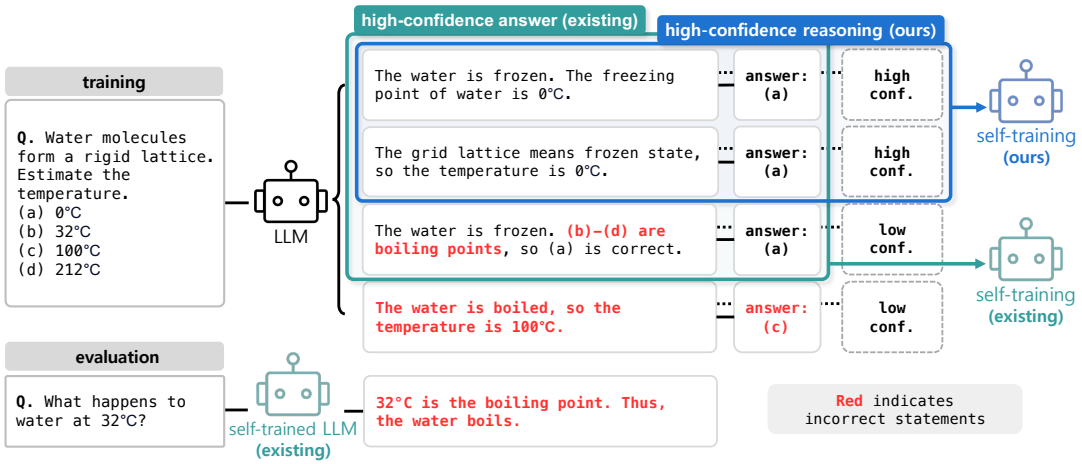

Figure 1: **Limitations in existing confidence-based self-training methods.** Existing self-training methods prefer reasoning paths associated with a high-confidence answer (a), estimated via majority voting. However, they fail to capture the errors in their third reasoning path, even though the answer is correct. As a result, they can degrade the reasoning capabilities of the LLM, e.g., preferring "(b)-(d) are boiling points" can lead to "32 °C is the boiling point", as shown in below. In contrast, our method measures reasoning-level confidence (as depicted by the dashed line) and fine-tunes LLMs to prefer high-confidence reasoning paths that yield correct statements.

To this end, we propose incorporating *reasoning-level confidence* into a confidence-based self-training method. As illustrated in Figure 1, our method evaluates the correctness of reasoning by estimating confidence in the reasoning paths rather than relying solely on answer-level confidence. This is motivated by Observation 2, which shows that outputs with higher reasoning-level confidence exhibit fewer errors, aligned with prior findings that such confidence is useful for identifying high-quality outputs (Becker and Soatto, 2024; Wan et al., 2025; Taubenfeld et al., 2025).

We then propose **CORE-PO**, a method that fine-tunes LLMs to prefer high-**CO**nfidence **RE**asoning paths using **P**olicy **O**ptimization. To be specific, we estimate the reasoning-level confidence using P(True) (Kadavath et al., 2022), measuring the probability that LLM returns "true" to the prompt asking whether the reasoning is correct. We consider the two ways to measure P(True) of reasoning: a monolithic way that assesses the entire reasoning path, and a statement-wise way that computes the average confidence across each step in the reasoning path. Then, we combine reasoning-level confidence with answer-level confidence, and fine-tune LLMs using direct preference optimization (Guo et al., 2024) to prefer high-confidence outputs.

In our experiments, we apply our self-training method to four arithmetic or scientific reasoning benchmarks: GSM8K (Cobbe et al., 2021), ARC-Challenge (Clark et al., 2018), GPQA (Rein et al., 2023), and MATH (Hendrycks et al., 2021). We also consider two external benchmarks, CRUXEval (Gu et al., 2024) and Game-of-24 (Lile, 2025), to evaluate the generalization capabilities on out-of-distribution tasks. Our method improves the accuracy of outputs on both in-distribution and out-of-distribution tasks by enhancing reasoning quality, compared to existing self-training approaches.

To conclude, our contributions can be summarized as follows:

- We identify a limitation of existing confidence-based self-training methods: they rely solely on *answer-level confidence*, which may fail to capture the errors in reasoning.
- We propose a new self-training method that incorporates *reasoning-level confidence* to better identify reasoning paths with fewer errors.
- Through extensive evaluation, we show that our method improves answer accuracy and reduces errors in reasoning compared to existing approaches on both in-distribution and out-of-distribution reasoning benchmarks.

2 Background: Self-Training LLMs

In this section, we describe the preliminaries of existing self-training methods for large language models (LLMs). We describe additional related works in Appendix A.

Notation. Let x denote a question provided to an LLM M_θ. For a given question x, we assume that the LLM outputs a sequence $s = [r, a]$, where r represents the reasoning path and a is the final

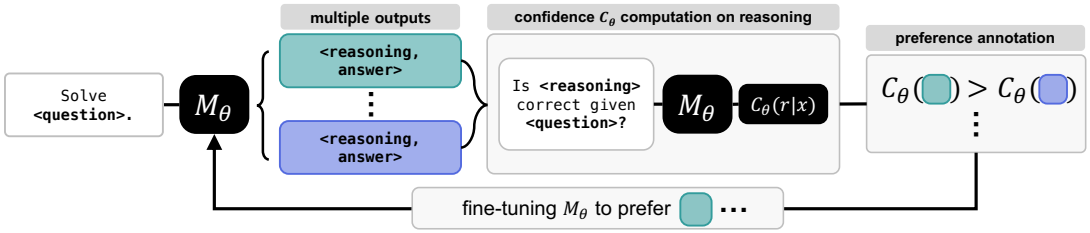

Figure 3: **Overview of CORE-PO.** The LLM M_θ generates multiple outputs, each consisting of a reasoning and an answer $s = [r, a]$ for a given question. Next, we measure the reasoning-level confidence $C_\theta(r|x) = \text{P(True)}$ for each reasoning path. Then, we fine-tune the LLM to prefer high-confidence reasoning paths.

answer induced from this reasoning path. In this paper, we are particularly interested in the model's confidence in its generated sequences. We denote the model's confidence score (or uncertainty estimate) on a given statement by $C_\theta(\cdot)$, which can be computed through various existing approaches such as self-consistency (Wang et al., 2023), semantic entropy (Kuhn et al.), or other uncertainty quantification techniques.

2.1 Existing works on self-training

Recent studies have shown that LLMs can self-improve through fine-tuning with pseudo-labels, e.g., the preference, generated by themselves. The key idea of these approaches is to transfer the performance gains from inference-time scaling methods such as self-consistency (Wang et al., 2023), tree-of-thoughts (Yao et al., 2023), or self-refinement (Madaan et al., 2023b), into training-time improvements through fine-tuning. These inference-time methods typically generate multiple candidate outputs and select high-quality outputs based on self-assessments, e.g., confidence estimation (Wang et al., 2023). Extending this approach, self-training methods fine-tune the LLMs to prefer outputs assessed as high-quality by the models themselves, leading to improved performance across various tasks (Huang et al., 2023; Yuan et al., 2024; Prasad et al., 2024; Zhang et al., 2024b; Zuo et al., 2025). Among these, several methods (Huang et al., 2023; Prasad et al., 2024; Zuo et al., 2025) leverage confidence scores, providing evidence that confidence-guided supervision can serve as a powerful training signal.

2.2 Existing works on confidence-based self-training

Recent self-training methods for LLMs aim to improve the reasoning capabilities by rewarding a reasoning path r that leads to an answer a with a high confidence score $C_\theta(a|x)$ (Huang et al., 2023; Prasad et al., 2024; Zhang et al., 2024b). These methods build on the observation that such an answer yields higher accuracy (Wang et al., 2023), and thus assume that the reasoning path r leading to such an answer is reliable. Specifically, they estimate the confidence score for an answer a using majority voting over multiple generated answers $a^1, \ldots, a^N$, i.e., $C_\theta(a|x) = \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}[a = a^i]$, following the concept of self-consistency (Wang et al., 2023). Next, they fine-tune the LLMs using reinforcement learning to prefer reasoning paths with high answer-level confidence $C_\theta(a \mid x)$.

3 Method

We introduce our confidence-based self-training method for large language models (LLMs) to improve their reasoning capabilities. First, we show (1) how existing confidence-based self-training methods can prefer incorrect reasoning paths, and (2) how incorporating reasoning-level confidence mitigates this issue (Section 3.1). Next, we describe a method that fine-tunes LLMs to prefer high-**CO**nfidence **RE**asoning paths using **P**olicy **O**ptimization, coined **CORE-PO** (Section 3.2).

3.1 Motivation for reasoning-level confidence in self-training

Our motivation stems from the limitations of confidence measures used for existing self-training methods (Huang et al., 2023; Prasad et al., 2024; Zhang et al., 2024b), which evaluate a reasoning–answer pair $[r, a]$ based on the confidence score on the answer $C_\theta(a|x)$. Here, we argue that such answer-level confidence may fail to capture the overall quality of the reasoning path (Figure 1 and Observation 1), as even an incorrect reasoning path may lead to a correct answer (Lanham et al., 2023; Zhang et al., 2024a). To remedy this, we advocate the use of reasoning-level confidence $C_\theta(r|x)$ as a way to evaluate the quality of reasoning paths (Observation 2).

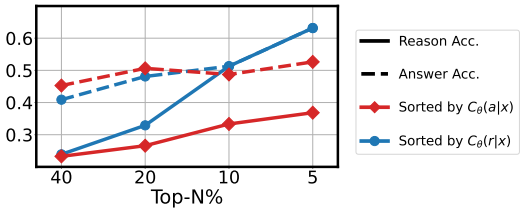

Figure 2: **Answer vs. reasoning accuracies.** We use Llama3.1-8B-Instruct (Meta AI, 2024). Reasoning-level accuracy coincides more closely with reasoning-level confidence than with answer-level confidence.

To support our claim, we conduct an observational experiment using multiple-choice questions in the GPQA dataset (Rein et al., 2023). We generate reasoning paths and assign two types of confidence scores to each path: one based on confidence in the final answer $C_\theta(a|x)$ (estimated via majority voting), and the other incorporating confidence in the reasoning path $C_\theta(r|x)$.[2] We describe detailed experimental settings in Appendix B.1. Then, we evaluate the correctness of the final answers and reasoning paths with accuracy.[3]

Our experiment makes the following observations, which support the use of reasoning-level confidence to evaluate the reasoning path.

Observation 1. Reasoning paths with high answer-level confidence are often incorrect, even when the final answers are correct. *See the gap between the answer-level accuracy (dashed red) and reasoning-level accuracy (solid red) for outputs with high answer-level confidence in Figure 2.*

Observation 2. High reasoning-level confidence $C_\theta(r|x)$ tends to yield accurate reasoning with fewer errors, coinciding with high answer-level accuracy. *See the accuracy of outputs with high reasoning-level confidence (blue) in Figure 2.*

The findings in Observation 1 hint at the pitfall of existing confidence-based self-training methods, that can train LLMs to prefer incorrect reasoning paths associated with high answer-level confidence by chance. Note that this pitfall also exists in conventional (not self-training) fine-tuning methods that use ground-truth answers and define the reward based on answer-level accuracy (Zelikman et al., 2022; Trung et al., 2024; DeepSeek-AI, 2025), as they may assign a positive reward to the incorrect reasoning path that leads to correct answer.[4]

[2]We actually consider $C_\theta(r, a|x) = C_\theta(r|x)C_\theta(a|x, r)$ for the later, using the measures described in Section 3.2.

[3]We use `o4-mini-2025-04-16` (OpenAI, 2025) to evaluate correctness of generated reasoning paths.

[4]See the ablation study in Section 4.3 for details.

Algorithm 1 Self-training LLMs with reasoning-level confidence scores

Input: An LLM M_θ, a set of questions $\mathcal{X}$
Initialize the reference model M_{ref} using M_θ
repeat
  Sample a question $x \sim \mathcal{X}$
  Sample $\{(r^i, a^i)\}_{i=1}^N \sim M_\theta(\cdot \mid x)$
  Compute $\{C_\theta(r^i, a^i \mid x)\}_{i=1}^N$
  Update θ to minimize $\mathcal{L}$ in Equation (1)
until convergence

In response, we propose incorporating reasoning-level confidence $C_\theta(r \mid x)$ to identify high-quality reasoning paths for self-training. While confidence scores have primarily been used for the factuality of a single statement, we highlight their utility in evaluating reasoning correctness, supported by Observation 2. Note that this observation also aligns with the finding in recent inference-time scaling methods (Becker and Soatto, 2024; Taubenfeld et al., 2025), which improve performance by selecting high-confidence reasoning paths.

3.2 CORE-PO: Self-training LLMs with reasoning-level confidence

We describe our self-training method, which learns to prefer high-confidence reasoning paths using policy optimization (CORE-PO). This method involves measuring reasoning-level confidence and training LLMs to prefer high-confidence reasoning paths. We provide an overview of our method in Figure 3 and Algorithm 1.

To measure the reasoning-level confidence $C_\theta(r|x)$, we use P(True) (Kadavath et al., 2022) which measures the probability that LLM returns "true" to the prompt asking whether the given reasoning r is correct for the given question x.[5] In detail, we consider two ways to measure confidence in a multi-statement reasoning path. First, for the monolithic P(True), we measure confidence in one-shot through asking the LLM to check the truthfulness of all the statements at once. Next, for the statement-wise P(True), we query the LLM for each statement to check the truthfulness, then average over the confidence scores, i.e., we measure $\frac{1}{T}\sum_{t=1}^{T} C_\theta(r_t|x, r_1, \ldots, r_{t-1})$ for multiple statements in the reasoning path $r = [r_1, \ldots, r_T]$.[6] We

[5]Note that **P(True)** has already shown promising results to evaluate the reasoning paths in existing inference-time scaling methods (Becker and Soatto, 2024; Taubenfeld et al., 2025)

[6]We compare both measures in Section 4.3.

Fine-tuning	Decoding	GSM8K	ARC-Challenge	GPQA^{ext}	MATH^{lv5}
Not Applied	Greedy	84.2	84.5	32.4	22.6
	Linguistic	85.7	86.0	31.8	22.1
	SC	89.6	86.6	34.3	25.6
	P(True$\|r, a$)	89.7	87.0	34.5	25.2
Learning from linguistic self-assessment of answer quality					
SR-PO (Kumar et al., 2024)	Greedy	85.2	86.2	34.3	19.8
	Linguistic	86.7	87.4	35.5	21.2
Learning to prefer reasoning paths with high answer-level confidence $C_\theta(a\|x)$					
SC-PO (Prasad et al., 2024)	Greedy	85.7	86.0	33.7	**25.1**
	SC	89.7	87.5	34.5	29.4
*Learning to prefer reasoning with high confidence scores $C_\theta(r\|x)$ **(ours)***					
CORE-PO (ours)	Greedy	**86.8**	**87.5**	**35.5**	24.6
	P(True$\|r, a$)	**90.5**	**89.2**	**36.1**	**29.8**

Table 1: **Performance of self-training methods on Llama3.1-8B-Instruct**. **Bold** indicates the best performance under greedy decoding or inference-time scaling methods (with sampling of eight outputs). Our method outperforms the considered baselines when applying both greedy decoding and inference-time scaling method.

also incorporate the answer-level confidence score $C_\theta(a|x, r)$, measured by P(True) given the question x and the reasoning path r. The implementations and prompts are described in Appendix B.2.

Next, we optimize the LLM M_θ to prefer reasoning–answer pairs with high confidence scores, measured as $C_\theta(a, r|x) = C_\theta(a|x, r)C_\theta(r|x)$. To this end, we use online Direct Preference Optimization (Guo et al., 2024, DPO), which samples two or more outputs for a given question and optimizes the LLM to assign higher likelihood to the output with higher confidence scores:

$$\mathcal{L} = \log \sigma \left(\beta \log \frac{M_\theta(s^l|x)}{M_{\text{ref}}(s^l|x)} - \beta \log \frac{M_\theta(s^w|x)}{M_{\text{ref}}(s^w|x)} \right) \quad (1)$$

where $s^w = [y^w, r^w]$ denotes a sequence with a higher confidence score than another sequence s^l, σ denotes the logistic function, and β is a hyperparameter in the DPO. The base reference model M_{ref} initializes M_θ. We fix the reference model during training without updates.

4 Experiments

In this section, we conduct experiments to validate **CORE-PO** across various reasoning tasks.

4.1 Experimental setup

In experiments, we consider two LLMs, Llama3.1-8B-Instruct (Meta AI, 2024) and Qwen2.5-7B-Instruct (Qwen Team, 2024), as base LLMs to implement our self-training method and baselines.

Tasks and datasets. We evaluate our self-training method on a range of reasoning tasks using the following datasets:

- **GSM8K** (Cobbe et al., 2021) consists of basic math questions requiring multi-step arithmetic reasoning. We use questions in training split for self-training and evaluate on the test split by the accuracy of the generated numerical answers.
- **ARC-Challenge** (Clark et al., 2018) contains multiple-choice science questions requiring commonsense reasoning. We use questions in the training split for self-training and evaluate on the test split by the accuracy of the selected choices.
- **GPQA** (Rein et al., 2023) contains graduate-level multiple-choice questions requiring advanced scientific reasoning. We use questions in GPQA-main and GPQA-extended splits for training and evaluation, respectively.
- **MATH** (Hendrycks et al., 2021) is a mathematic reasoning benchmark, which consists of challenging high-school math problems. We use questions in the training split for self-training and evaluate on Level-5 questions in the test split.

In addition, we evaluate the out-of-domain generalization capabilities of self-trained LLMs using the following two benchmarks:

- **CRUX** (Gu et al., 2024) is a benchmark for evaluating code understanding and execution. We use tasks of predicting the output of Python functions given inputs, i.e., CRUX^{out}.

Fine-tuning	Decoding	GSM8K	ARC-Challenge	QPQA^{ext}	MATH^{lv5}
Not Applied	Greedy	90.0	89.1	30.6	45.4
	Linguistic	91.1	89.9	31.4	47.9
	SC	92.0	91.5	33.7	54.6
	P(True$\|r, a$)	93.2	91.3	34.1	55.0
Learning from linguistic self-assessment of answer quality					
SR-PO (Kumar et al., 2024)	Greedy	90.9	90.2	32.6	48.1
	Linguistic	92.6	91.3	35.8	49.3
Learning to prefer reasoning with high answer-level confidence $C_\theta(a\|x)$					
SC-PO (Prasad et al., 2024)	Greedy	91.0	91.0	34.3	**49.6**
	SC	93.0	92.0	36.3	55.7
Learning to prefer reasoning with high reasoning-level confidence $C_\theta(r\|x)$ ***(ours)***					
CORE-PO (ours)	Greedy	**91.3**	**92.2**	**37.5**	**49.6**
	P(True$\|r, a$)	**93.5**	**92.8**	**38.5**	**55.8**

Table 2: **Performance of self-training methods on Qwen2.5-7B-Instruct. Bold** indicates the best performance under greedy decoding or inference-time scaling methods (with sampling of eight outputs). Our method outperforms the considered baselines when applying both greedy decoding and inference-time scaling method.

- **Game of 24** (Lile, 2025) is a reasoning benchmark, where the goal is to determine whether a given set of four integers can be combined using operations (addition, subtraction, multiplication, or division) to induce the number 24.

We provide detailed data statistics of the above datasets in Appendix B.3.

Baselines. We compare our CORE-PO with existing self-training approaches. We first consider self-rewarding-based preference optimization (Kumar et al., 2024, SR-PO), which trains LLMs using linguistic self-assessments of answer quality, e.g., assigning higher scores to outputs involving expert-level knowledge. Next, we consider self-consistency preference optimization (Zhang et al., 2024b, SC-PO), which trains LLMs to prefer outputs with high answer-level confidence scores $C_\theta(a|x)$ measured via self-consistency score (Wang et al., 2023, SC), i.e., majority voting over multiple sampled outputs.

To compare the performance, we generate outputs from each fine-tuned and base LLM using two decoding schemes: (1) greedy decoding and (2) inference-time scaling methods. For (2), we generate multiple outputs and select the most promising one from the self-assessment score associated with each self-training method: the linguistic self-assessment score (Linguistic) for SR-PO, the majority-voting score over multiple answers (SC) for SC-PO, and the estimated confidence score on the reasoning-answer pair (P(True$|r, a$)) for our self-training method.

Implementations. In our experiments, we apply self-training methods to LLMs using unified training question sets from the aforementioned datasets. Here, we implement our self-training method using monolithic P(True) to measure confidence scores on reasoning paths.[7] We then evaluate the self-trained LLMs on test question sets from both in-distribution and out-of-distribution datasets. For training, we generate $N = 5$ outputs for each question with $T = 1.0$ temperature and top-$p = 0.9$ (Holtzman et al., 2020). We also apply a low rank adaptation (Hu et al., 2022) with rank 128 and $\alpha = 256$ to both Llama3.1–8B–Instruct and Qwen2.5–7B–Instruct. We also use $\beta = 0.1$ in direct preference optimization. To apply inference-time scaling methods, we randomly sample eight outputs with $T = 0.7$ and top-$p = 0.9$. We further describe detailed experimental settings and prompts in Appendix B.

4.2 Main results

Competitive performance for self-training. We present the results in Tables 1 and 2, which are obtained from our implementations using Llama3.1-8B-Instruct and Qwen2.5-7B-Instruct, respectively. One can observe that our self-trained LLMs outperform the base LLMs and the LLMs self-trained with SR-PO and SC-PO algorithms. In particular, one can observe that self-training with reasoning-level confidence (CORE-PO) yields larger gains

[7] We present the results of self-training with statement-wise P(True) in Section 4.3.

Fine-tuning	Decoding	GSM8K		ARC-Challenge	
		Conf. Score	Reason Acc.	Conf. Score	Reason Acc.
Not Applied	Greedy	0.89	84.2	0.84	79.2
	P(True|r,a)	0.98	89.7	0.94	81.2
CORE-PO (ours)	Greedy	**0.94**	**86.8**	**0.95**	**81.5**
	P(True|r,a)	**0.99**	**90.4**	**0.99**	**84.9**

Table 3: **Reasoning-level confidence and accuracy.** The base LLM is Llama3.1-8B-Instruct. **Bold** indicates the best performance under greedy decoding or inference-time scaling methods (with sampling of eight outputs). Our self-training method enables the LLM to improve reasoning-level confidence and accuracy.

Fine-tuning	Decoding	CRUX^{out}	Game of 24
Not Applied	Greedy	34.8	7.2
	Linguistic	34.7	13.2
	SC	46.1	15.3
	P(True|r,a)	41.0	21.0
SR-PO	Greedy	31.6	8.8
	Linguistic	36.2	10.5
SC-PO	Greedy	43.8	8.3
	SC	**50.0**	11.9
CORE-PO	Greedy	**47.1**	**18.8**
	P(True|r,a)	48.0	**22.1**

Table 4: **Performance of self-training methods on out-of-distribution tasks.** The base LLM is Llama3.1-8B-Instruct. **Bold** indicates the best performance under greedy decoding or inference-time scaling methods (with sampling of eight outputs). Our self-training method shows competitive performance or yields best results compared to considered baselines.

over self-training with answer-level confidence (SC-PO) on the ARC-Challenge and GPQA benchmarks, whose multiple-choice format often allows incorrect reasoning paths to lead to high-confidence answers by chance. Furthermore, on these benchmarks, one can see that our fine-tuned LLMs achieve higher accuracy with greedy decoding than the base LLMs using inference-time scaling. These overall results highlight that our self-training method enables the LLMs to generate higher-quality answers.

Improved reasoning-level accuracy and confidence in reasoning. In addition, we also measure the confidence score on the generated reasoning paths and the reasoning-level accuracy.[8] We present the results in Table 3. Here, one can observe that our fine-tuned model increases reasoning-level confidence compared to the base LLM, which coincides with an increase in reasoning-level accuracy. These results also provide evidence that our self-training method enhances the reasoning capabilities of the base LLM by preferring reasoning paths with higher confidence.

Generalization to out-of-distribution tasks. We also conduct validation on out-of-distribution tasks. We present the results in Table 4. One can observe that our method yields significant improvements on both CRUX^{out} and Game of 24, compared to the base LLM. In particular, one can see that our self-training method yields the best performance on Game of 24 when applying both greedy decoding and inference-time scaling. While SC-PO shows competitive performance on CRUX^{out}, it shows limited improvement on Game of 24. These results highlight that our method generalizes better to out-of-distribution tasks than existing baselines.

4.3 Ablation studies

Monolithic P(True) vs. statement-wise P(True). We also conduct experiments by implementing our method using statement-wise P(True). We present the results in Table 5. Here, one can see that neither method consistently outperforms the other: self-training with statement-wise P(True) yields high performance on the GSM8k, ARC-Challenge, and MATH^{lv5} benchmarks, but yields relatively low performance on the GPQA^{ext} benchmark. Nevertheless, both methods consistently outperform the base LLM. These results highlight that the performance improvements of our method do not stem from a particular implementation of confidence estimation, but from the philosophy of preferring high reasoning-level confidence.

Fine-tuning with ground-truth answers. We also conduct experiments by incorporating reasoning-

[8] We evaluate the reasoning-level accuracy using external oracle `o4-mini-2025-04-16` (OpenAI, 2025) by querying whether the reasoning is correct.

Fine-tuning	GSM8K	ARC-Challenge	GPQA^{ext}	MATH^{Lv5}
Not Applied	84.2	84.5	32.4	22.6
CORE-PO w/ Monolithic P(True)	86.8	87.5	35.5	24.7
CORE-PO w/ Statement-wise P(True)	88.5	88.0	34.1	25.3

Table 5: **Self-training with two different confidence measures.** The base LLM is Llama3.1-8B-Instruct. We use greedy decoding for the comparison. Both monolithic P(True) and statement-wise P(True) consistently improve the base model, but neither variant shows a clear advantage over the other.

Reward signal	Decoding	Answer Acc.	Reason Acc.
Learning to prefer reasoning paths leading to correct answers			
Answer Acc.	Greedy	87.4	73.6
	P(True|r,a)	88.1	78.1
Learning to prefer high-confidence reasoning paths leading to correct answers ***(ours)***			
Answer Acc. + Reason Conf. (ours)	Greedy	**88.3**	**81.9**
	P(True|r,a)	**90.1**	**85.6**

Table 6: **Experiments with ground-truth answers.** The base LLM is Llama3.1-8B-Instruct. **Bold** indicates the best performance under greedy decoding or inference-time scaling methods (with sampling of eight outputs). We use the training and test splits of the ARC-Challenge dataset (Clark et al., 2018). Incorporating reasoning-level confidence $C_\theta(r|x)$ leads to improvements in both answer-level and reasoning-level accuracy.

level confidence into a conventional fine-tuning (not self-training) method, which uses ground-truth answers and defines answer-level accuracy as the reward (Zelikman et al., 2022; Trung et al., 2024; DeepSeek-AI, 2025). This experiment is motivated by the following pitfall in this method: rewarding incorrect reasoning paths that yield correct answers by chance. Motivated by this issue, we hypothesize that incorporating reasoning-level confidence can prevent the LLM from preferring incorrect reasoning paths that lead to correct answers. Specifically, we compare two fine-tuning methods: one that assigns rewards based solely on answer-level accuracy, and another that incorporates both answer-level accuracy and reasoning-level confidence.

In Table 6, we present the results of experiments conducted on the multiple-choice questions from the ARC-Challenge dataset. One can see that incorporating reasoning-level confidence improves both answer-level and reasoning-level accuracies. Notably, the LLM fine-tuned solely on answer-level accuracy shows significantly lower reasoning-level accuracy compared to its answer-level accuracy. We observe that this degradation stems from the model's reasoning patterns, which tend to generate unnecessarily long reasoning paths by evaluating all options in the multiple-choice question. Here, the LLM often makes mistakes in evaluating some of the choices, even when the final answer is correct. Table 7 presents examples of these patterns.

Question
At which temperature does water freeze? [I] 0°C, [II] 32°C, [III] 100°C, [IV] 212°C
Fine-tuning with Answer Acc.
[I] is correct: water freezes at 0°C. [II] is incorrect: 32 is boiling point in Fahrenheit. [III] is incorrect: 100°C is the boiling point of water. [IV] is incorrect: 212 is the boiling point in Fahrenheit. **Answer: [I]**
Fine-tuning with Answer Acc. + Reason Conf. (ours)
The question asks for the freezing point of water. In the Celsius scale, this is 0°C, a well-known scientific fact across disciplines such as chemistry and physics. **Answer: [I]**

Table 7: **Example of generated reasoning paths.** The reasoning content is summarized due to its excessive length. The first reasoning path involves errors despite leading to the correct answer.

5 Conclusion

In this paper, we propose a new confidence-based self-training method that addresses a key limitation of existing approaches: the exclusive reliance on answer-level confidence, which does not capture the overall quality of the reasoning. By incorporating reasoning-level confidence, our method fine-tunes LLMs to prefer high-confidence reasoning paths with fewer errors, thereby improving their reasoning capabilities. Empirical results on six benchmarks show that our method improves the reasoning capabilities of LLMs on both in-distribution and out-of-distribution tasks, outperforming existing self-training methods.

Limitations

Confidence measures. Although we use a confidence measure that is relatively reliable than pure likelihoods over generated sequences, it can suffer from overconfidence due to the inherent calibration issues of large language models. This still poses the risk of reinforcing incorrect reasoning paths that are assigned high confidence scores due to miscalibrated confidence estimates. Next, our evaluation considers confidence metrics based solely on P(True), but incorporating alternative measures, e.g., semantic entropy (Kuhn et al.) or contextualized likelihood (Lin et al., 2024), may provide a more robust estimation of the confidence. An interesting avenue for future work is to develop and incorporate more robust and well-calibrated confidence measures into our method.

Language-specific experiments. Our experiment focuses exclusively on English, and we do not explore the applicability of our method to other languages, e.g., morphologically rich or typologically diverse languages. Since reasoning pattern and confidence calibration can vary significantly across languages due to linguistic structure and pretraining data distribution, it remains unexplored whether our findings generalize beyond English.

Prompting. We evaluate our method in a zero-shot setting using the default system prompt, i.e., "Be a helpful assistant.". However, more advanced prompting strategies, such as few-shot prompting or task-specific system prompts (Brown et al., 2020), may further improve performance.

Human evaluation. In this paper, we do not conduct human evaluation to assess the quality or faithfulness of the generated outputs, leaving open the question of alignment with human judgment.

Experiments on larger-scale models. Our experiments only consider large language models with up to 7.5B or 8B parameters due to the limited computational budgets. The generalizability of our method to larger models (e.g., 70B) remains unexplored and is left for future work.

References

Evan Becker and Stefano Soatto. 2024. Cycles of thought: Measuring llm confidence through stable explanations. *Preprint*, arXiv:2406.03441.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33:1877–1901.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *Preprint*, arXiv:2110.14168.

DeepSeek-AI. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Alex Gu, Baptiste Roziere, Hugh James Leather, Armando Solar-Lezama, Gabriel Synnaeve, and Sida Wang. 2024. CRUXEval: A benchmark for code reasoning, understanding and execution. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 16568–16621. PMLR.

Sylvain Gugger, Lysandre Debut, Thomas Wolf, Philipp Schmid, Zachary Mueller, Sourab Mangrulkar, Marc Sun, and Benjamin Bossan. 2022. Accelerate: Training and inference at scale made simple, efficient and adaptable. https://github.com/huggingface/accelerate.

Shangmin Guo, Biao Zhang, Tianlin Liu, Tianqi Liu, Misha Khalman, Felipe Llinares, Alexandre Rame, Thomas Mesnard, Yao Zhao, Bilal Piot, and 1 others. 2024. Direct language model alignment from online ai feedback. *arXiv preprint arXiv:2402.04792*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. In *International Conference on Learning Representations*.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Jiaxin Huang, Shixiang Gu, Le Hou, Yuexin Wu, Xuezhi Wang, Hongkun Yu, and Jiawei Han. 2023. Large language models can self-improve. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 1051–1068, Singapore. Association for Computational Linguistics.

Fangkai Jiao, Chengwei Qin, Zhengyuan Liu, Nancy Chen, and Shafiq Joty. 2024. Learning planning-based reasoning by trajectories collection and process

reward synthesizing. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 334–350.

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, Scott Johnston, Sheer El-Showk, Andy Jones, Nelson Elhage, Tristan Hume, Anna Chen, Yuntao Bai, Sam Bowman, Stanislav Fort, and 17 others. 2022. Language models (mostly) know what they know. *Preprint*, arXiv:2207.05221.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213.

Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation. In *The Eleventh International Conference on Learning Representations*.

Abhishek Kumar, Robert Morabito, Sanzhar Umbet, Jad Kabbara, and Ali Emami. 2024. Confidence under the hood: An investigation into the confidence-probability alignment in large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 315–334, Bangkok, Thailand. Association for Computational Linguistics.

Tamera Lanham, Anna Chen, Ansh Radhakrishnan, Benoit Steiner, Carson Denison, Danny Hernandez, Dustin Li, Esin Durmus, Evan Hubinger, Jackson Kernion, and 1 others. 2023. Measuring faithfulness in chain-of-thought reasoning. *arXiv preprint arXiv:2307.13702*.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. Let's verify step by step. In *International Conference on Learning Representations (ICLR)*.

Nick Lile. 2025. Game of 24: A benchmark for arithmetic reasoning. https://huggingface.co/datasets/nlile/24-game. Accessed: 2025-05-07.

Zhen Lin, Shubhendu Trivedi, and Jimeng Sun. 2024. Contextualized sequence likelihood: Enhanced confidence scores for natural language generation. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 10351–10368, Miami, Florida, USA. Association for Computational Linguistics.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023a. Self-refine: Iterative refinement with self-feedback. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, and 1 others. 2023b. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36:46534–46594.

Meta AI. 2024. Introducing llama 3.1: Our most capable models to date. https://ai.meta.com/blog/meta-llama-3-1/. Accessed: 2025-05-19.

OpenAI. 2025. Introducing openai o3 and o4-mini. https://openai.com/index/introducing-o3-and-o4-mini/. Accessed: 2025-05-15.

Archiki Prasad, Weizhe Yuan, Richard Yuanzhe Pang, Jing Xu, Maryam Fazel-Zarandi, Mohit Bansal, Sainbayar Sukhbaatar, Jason Weston, and Jane Yu. 2024. Self-consistency preference optimization. *arXiv preprint arXiv:2411.04109*.

Qwen Team. 2024. Qwen2.5: A party of foundation models. https://huggingface.co/Qwen/Qwen2.5-7B-Instruct. Accessed: 2025-05-07.

Leonardo Ranaldi and Andrè Freitas. 2024. Self-refine instruction-tuning for aligning reasoning in language models. *arXiv preprint arXiv:2405.00402*.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. 2023. Gpqa: A graduate-level google-proof q&a benchmark. *arXiv preprint arXiv:2311.12022*.

Amir Taubenfeld, Tom Sheffer, Eran Ofek, Amir Feder, Ariel Goldstein, Zorik Gekhman, and Gal Yona. 2025. Confidence improves self-consistency in llms. *arXiv preprint arXiv:2502.06233*.

Katherine Tian, Eric Mitchell, Allan Zhou, Archit Sharma, Rafael Rafailov, Huaxiu Yao, Chelsea Finn, and Christopher Manning. 2023. Just ask for calibration: Strategies for eliciting calibrated confidence scores from language models fine-tuned with human feedback. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5433–5442, Singapore. Association for Computational Linguistics.

Luong Trung, Xinbo Zhang, Zhanming Jie, Peng Sun, Xiaoran Jin, and Hang Li. 2024. ReFT: Reasoning with reinforced fine-tuning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7601–7614, Bangkok, Thailand. Association for Computational Linguistics.

Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, Shengyi Huang, Kashif Rasul, and Quentin Gallouédec. 2020. Trl: Transformer reinforcement learning. https://github.com/huggingface/trl.

Guangya Wan, Yuqi Wu, Jie Chen, and Sheng Li. 2025. Reasoning aware self-consistency: Leveraging reasoning paths for efficient LLM sampling. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 3613–3635, Albuquerque, New Mexico. Association for Computational Linguistics.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, and 3 others. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

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

Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Xian Li, Sainbayar Sukhbaatar, Jing Xu, and Jason Weston. 2024. Self-rewarding language models. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*. OpenReview.net.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. 2022. STar: Bootstrapping reasoning with reasoning. In *Advances in Neural Information Processing Systems*.

Dan Zhang, Sining Zhoubian, Ziniu Hu, Yisong Yue, Yuxiao Dong, and Jie Tang. 2024a. ReST-MCTS*: LLM self-training via process reward guided tree search. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Xiaoying Zhang, Baolin Peng, Ye Tian, Jingyan Zhou, Lifeng Jin, Linfeng Song, Haitao Mi, and Helen Meng. 2024b. Self-alignment for factuality: Mitigating hallucinations in LLMs via self-evaluation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1946–1965, Bangkok, Thailand. Association for Computational Linguistics.

Xuan Zhang, Chao Du, Tianyu Pang, Qian Liu, Wei Gao, and Min Lin. 2024c. Chain of preference optimization: Improving chain-of-thought reasoning in LLMs. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. 2023. Automatic chain of thought prompting in large language models. In *The Eleventh International Conference on Learning Representations*.

Andrew Zhao, Yiran Wu, Yang Yue, Tong Wu, Quentin Xu, Yang Yue, Matthieu Lin, Shenzhi Wang, Qingyun Wu, Zilong Zheng, and Gao Huang. 2025. Absolute zero: Reinforced self-play reasoning with zero data. *Preprint*, arXiv:2505.03335.

Yuxin Zuo, Kaiyan Zhang, Shang Qu, Li Sheng, Xuekai Zhu, Biqing Qi, Youbang Sun, Ganqu Cui, Ning Ding, and Bowen Zhou. 2025. Ttrl: Test-time reinforcement learning. *Preprint*, arXiv:2504.16084.

A Related works

Confidence measures for LLMs. Large language models (LLMs) often generate incorrect outputs due to hallucinations, which highlights the importance of estimating confidence in their outputs. To this end, several methods have been proposed, including self-consistency (Wang et al., 2023), semantic entropy over semantically equivalent sequences (Kuhn et al.), the probability of truth P(True) (Kadavath et al., 2022), or asking the model to express its confidence in linguistic form (Tian et al., 2023). In addition, Lin et al. (2024) propose computing confidence using the likelihoods of important tokens that determine the semantics of the sequence. Although such confidence measures have mainly been applied to single-statement factuality checks, recent studies have shown that measures based on P(True) can be utilized to estimate the confidence in the reasoning path (Becker and Soatto, 2024; Taubenfeld et al., 2025).

Inference-time scaling methods for LLMs. Inference-time scaling methods improve the quality of LLM outputs by self-assessing multiple generated outputs. Among them, the self-consistency-based method (Wang et al., 2023) selects the most frequent answer obtained through majority voting. Other approaches include tree-of-thoughts (Yao et al., 2023), which expands the search space over intermediate steps, and self-refinement inference (Madaan et al., 2023b), which iteratively refines outputs using the LLM itself. In addition, Taubenfeld et al. (2025); Wan et al. (2025) recently proposed incorporating confidence scores on reasoning paths into the self-consistency method and showed notable performance improvements.

Training of reasoning for LLMs. To enhance reasoning abilities, LLMs are initially fine-tuned using various supervision signals. A straightforward approach is supervised fine-tuning on high-quality reasoning datasets (Cobbe et al., 2021; Trung et al., 2024), or direct preference optimization on reasoning datasets annotated with human preferences (Meta AI, 2024). While effective, collecting such datasets is costly. As an alternative approach, several studies instead consider reinforcement learning methods that rely solely on ground-truth answers, using answer-level accuracy as the reward signal (Zelikman et al., 2022; Trung et al., 2024; DeepSeek-AI, 2025). In addition, several studies propose training process reward models (PRMs) that assess the quality of intermediate reasoning steps. Lightman et al. (2024) and Jiao et al. (2024) train PRMs using human-annotated preferences on individual reasoning steps and answer-level ground-truth labels, respectively.

Other self-training approaches for LLMs. We further describe self-training methods derived from inference-time scaling techniques that are not based on confidence. First, Kumar et al. (2024) propose using linguistic assessments of output quality, e.g., assigning high scores to outputs that exhibit expert-level knowledge, as reward signals for fine-tuning LLMs. Next, Ranaldi and Freitas (2024) propose using outputs obtained from self-refinement inference (Madaan et al., 2023b), as these outputs typically exhibit higher quality than the initial outputs. Lastly, Zhang et al. (2024c) leverage preference signals over intermediate reasoning steps derived from tree-of-thoughts inference (Yao et al., 2023). A concurrent line of work, Absolute Zero (Zhao et al., 2025) shows that LLMs can improve their reasoning abilities by self-generating and solving code-based tasks, without relying on external data or human supervision.

B Experimental details

We use all datasets and models in accordance with their intended use for academic research, following their respective licenses.

B.1 Observational experiment

For the observational experiment, we use questions from the GPQA-main dataset (Rein et al., 2023). For each question, we generate an output consisting of a reasoning path and a final answer using Llama3.1-8B-Instruct (Meta AI, 2024). To be specific, to obtain reasoning paths with high answer-level or reasoning-level confidence, we generate 16 outputs per question and select the one with the highest confidence score. As a result, we obtain a triplet (question, reasoning path with high answer-level confidence, reasoning path with high reasoning-level confidence) for each question in the GPQA-main dataset. We then evaluate the correctness of each reasoning path using an external tool: `o4-mini-2025-04-16`. Before evaluating the correctness of each reasoning path using `o4-mini-2025-04-16`, we first consider the reasoning to be incorrect if the answer is incorrect.

B.2 Prompts

Prompt for solving ARC-Challenge and GPQA. We use the following prompt to solve the given multiple-choice question [question].

> Answer the following question using **reasoning** before providing a final answer. Provide a precise, structured, and well-reasoned response.
>
> **Question:** [question]
>
> ### Response Format
>
> **Understanding the question:** <identify key details>
>
> **Reasoning:** <perform chain-of-thought>
>
> **Final answer:** "The answer is <choose the most promising single answer from [I] / [II] / [III] / [IV]> which is <copy the content>"
>
> Ensure correctness and clarity. Return a concise and definitive response to the question. DO NOT RETURN TWO OR MORE ANSWERS. STRICTLY FOLLOW THE RESPONSE FORMAT.

Prompt for solving GSM8K and MATH. We use the following prompt to solve the given numeric-response question [question].

> Answer the following question using **reasoning** before providing a final answer. Provide a precise, structured, and well-reasoned response.
>
> **Question:** [question]
>
> ### Response Format
>
> **Understanding the question:** <identify key details>
>
> **Reasoning:** <perform chain-of-thought>
>
> **Final answer:** "The answer is $<value>$"
>
> Ensure correctness and clarity. Return a concise and definitive response to the question. STRICTLY FOLLOW THE RESPONSE FORMAT.

Prompts for solving questions in Game of 24. We use the following prompt to complete the expression given the four digit numbers [four digits].

> Answer the following question using **reasoning** before providing a final answer. Provide a precise, structured, and well-reasoned response.
>
> **Question:** "Write an equation using basic arithmetic operations (+ - * /) to obtain 24 from the four given numbers, e.g., "(4 + 8) * (6 - 4) = 24" from the input [4, 4, 6, 8]. You must use all the given numbers exactly once, i.e., simply rearrange them. Do not use any additional numbers. Parentheses can be used to control the order of operations. Now, write an expression using exactly the given numbers [four digits] that results in 24.
>
> ### Response Format
>
> **Understanding the question:** <identify key details>
>
> **Reasoning:** <perform chain-of-thought>
>
> **Final answer:** "The answer is $<value>$"
>
> Ensure correctness and clarity. Return a concise and definitive response to the question. THE LHS EXPRESSION MUST USE THE FOUR GIVEN NUMBERS EXACTLY ONCE. DO NOT SIMPLIFY THE FINAL EQUATION. STRICTLY FOLLOW THE RESPONSE FORMAT.

Prompt for solving Crux$^{\text{out}}$. We use the following prompt to predict the output given the code [code] and the input [input].

> You are given a Python function and an assertion containing an input to the function. Complete the assertion with a literal (no unsimplified expressions, no function calls) containing the output when executing the provided code on the given input, even if the function is incorrect or incomplete. Execute the program step by step as **reasoning** before providing a final answer. Provide a precise, structured, and well-reasoned response.
>
> **Code:** [code]
>
> ### Response Format
>
> **Reasoning:** <perform chain-of-thought (step-by-step execution)>
>
> **Final answer:** assert f([input]) == <output>"""
>
> Ensure correctness and clarity. Return a concise and definitive response to the question. STRICTLY FOLLOW THE RESPONSE FORMAT.

Prompt for confidence estimation. We use the following prompts to estimate the confidence score on reasoning path [reasoning] and answer [answer]. As we obtain multiple reasoning paths in inference-time scaling, we also augment additional reasoning paths [example i] for $i = 1, \ldots, M$ in estimating the confidence score on the reasoning path [reasoning], where $M = 4$.

Measuring monolithic $C_\theta(r|x)$

Answer whether the **selected reasoning** is correct for the given **question**. Additionally, we provide randomly generated reasoning before presenting the selected reasoning.

Question: [question]

Randomly generated reasoning 1 (this may be either correct or incorrect): [example 1]

...

Randomly generated reasoning M (this may be either correct or incorrect): [example M]

Selected reasoning: [reasoning]

Is the **selected reasoning** correct?

A) True

B) False

The **selected reasoning** is: [A / B, depending on whether the **selected reasoning** is correct given the **question**]

Measuring statement-wise $C_\theta(r_k|x, r_1, \ldots, r_{k-1})$

Answer whether the **new reasoning statement** is correct for the given **previous reasoning statements** and the **question**.

Question: [question]

Previous reasoning statements:
[step-1]
...
[step-$(k-1)$]

New reasoning statement: [step-k]

Is the **new reasoning statement** correct?

A) True

B) False

The **new reasoning statement** is: [A / B, depending on whether the **new reasoning statement** is correct given the **previous reasoning statements** and the **question**]

Measuring $C_\theta(a|r, x)$

Answer whether the **selected answer** is correct for the given **question**, based on the provided **reasoning**.

Question: [question]

Reasoning: [reasoning]

Selected answer: [answer]

Is the **selected answer** correct?

A) True

B) False

The **selected answer** is: [A / B, depending on whether the **selected answer** is correct given the **question** and the **reasoning**]

Evaluating reasoning correctness. We use the following prompt to evaluate the reasoning [reasoning] given the question [question].

Given a question, answer whether the reasoning could be correct.

Respond ONLY in JSON format:

{
"verdict": "correct" or "incorrect"
}

Question: [question]

Reasoning: [reasoning]

B.3 Data statistics

We provide detailed data statistics for the datasets used in training and evaluation. The training and test splits of GSM8k dataset (Cobbe et al., 2021) contain 7.4k and 1.3k questions, respectively. The ARC-Challenge dataset (Clark et al., 2018) includes training, validation, and test splits, containing 1.1k, 0.3k, and 1.1k questions, respectively. For the GPQA dataset (Rein et al., 2023) (involving 0.4k and 0.5k questions in main and extended splits), we use questions with lengths below $1,280$, where the resulting main and extended splits include 420 and 509 questions, respectively. The MATH dataset (Hendrycks et al., 2021) contains 7.5k training questions and 0.7k Level-5 test questions. The CRUXEval (Guo et al., 2024) and Game of 24 dataset (Lile, 2025) contain 0.8k and 1.3k questions, respectively.[9]

[9]We further clarify that GSM8K, MATH, 24-Game, and CruxEval are released under open-source licenses (Apache 2.0 or MIT). GPQA and ARC-Challenge are distributed under the CC-BY-4.0 license.

B.4 Implementations

We use four NVIDIA A100 SXM4 80GB GPUs. We save checkpoints every 200 steps and select the model with the highest accuracy on the ARC-Challenge validation split. Training the selected model typically takes two to four days. We apply hyper-parameter searching for learning rate over $\{1\text{e-}6, 5\text{e-}6\}$. We also apply a low rank adaptation (Hu et al., 2022) with rank 128 and $\alpha = 256$. At each gradient step, gradient clipping with a maximum norm of 1.0 is applied. We report results from a single run.

- For CORE-PO (ours), which uses online DPO, we generate $N = 5$ outputs for each question in the training set. We construct preference pairs of outputs by evaluating their confidence measures, as described in Section 3.2. The detailed prompts are provided in Appendix B.2.
- For SR-PO (Kumar et al., 2024) which uses offline DPO, we generate $N = 5$ outputs for each question in the training set. Then, we construct preference pairs by evaluating their scores using original self-rewarding prompts of SR-PO. In addition, we consider multiple iterations proposed in this method (Kumar et al., 2024), where each iteration involves an update of the reference model in DPO. We conduct two iterations in our experimental setup, where the LLM achieving the highest validation accuracy is selected.
- For SC-PO (Prasad et al., 2024) which uses offline DPO. This method samples a larger number of outputs ($N = 8$) for each question, since preference pairs are constructed only when the majority voting scores over answers differ by at least 3 (Prasad et al., 2024). A smaller number of outputs ($N = 5$) often fails to construct preference pairs under this criterion. In addition, we consider multiple iterations proposed in this method (Prasad et al., 2024), where each iteration involves an update of the reference model in DPO. We conduct two iterations in our experimental setup, where the LLM achieving the highest validation accuracy is selected.

We further clarify that our implementations are based on the `transformers` library (Wolf et al., 2020), the `trl` library (von Werra et al., 2020), and the `accelerate` library (Gugger et al., 2022).[10]

[10]We use Qwen2.5-7B (Apache 2.0) and LLaMA 3.1-8B (LLaMA 3.1 Community License), both of which allow use and redistribution under their respective terms.

C Use of AI assistants

We used AI-based writing assistants to improve the grammar. These tools were used only for editorial improvements. The technical content, methodology, and experimental results were entirely authored by the researchers.