Verbalized Confidence Triggers Self-Verification
: Emergent Behavior Without Explicit Reasoning Supervision

Anonymous Authors[1]

Abstract

Uncertainty calibration is essential for the safe deployment of large language models (LLMs), particularly when users rely on verbalized confidence estimates. While prior work has focused on classifiers or short-form generation, confidence calibration for chain-of-thought (CoT) reasoning remains largely unexplored. Surprisingly, we find that supervised fine-tuning with scalar confidence labels alone suffices to elicit self-verification behavior of language models, without any explicit reasoning supervision or reinforcement learning-based rewards. Despite being trained only to produce a verbalized confidence score without any self-verifying examples, the model learns to generate longer and self-checking responses for low-confidence queries while providing more concise answers for high-confidence ones. We further propose a simple rethinking method that boosts performance via test-time scaling based on calibrated uncertainty. Experiments on GSM8K and held-out reasoning tasks such as MATH-500 and ARC-Challenge show that our confidence-aware fine-tuning improves both calibration and accuracy, while also enhancing interpretability by aligning the model's reasoning path with its confidence.

1. Introduction

Large language models (LLMs) demonstrate strong performance not only in natural language generation but also in complex reasoning and decision-support tasks across diverse domains (Achiam et al., 2023; Guo et al., 2025). Their application to high-stakes settings such as medical diagnosis and personalized financial analysis has drawn increasing attention (Goh et al., 2024; Qiu et al., 2024; Takayanagi et al., 2025), with the potential to reduce expert workload and accelerate decision-making.

Nonetheless, ensuring the reliability of LLMs remains a critical challenge. Models frequently produce incorrect outputs with high confidence, and such overconfident errors can lead to harmful decisions when left undetected by users (K. Zhou et al., 2024). These hallucinations pose risks that go beyond factual inaccuracies, with implications for healthcare, law, and finance (X. Du et al., 2024). To mitigate these risks, LLMs must be able to quantify and communicate their uncertainty in a human interpretable manner, such as through *verbalized confidence*.

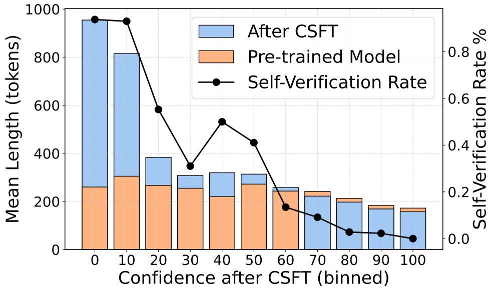


Figure 1: Generation length and self-verification rate across confidence bins on GSM8K using the CSFT-trained `LLaMA-3.2-3B-Instruct` model. Lower-confidence bins yield longer outputs and higher self-verification rates, suggesting a learned fallback behavior. Overall, 20% of generations showed self-verification (measured using `GPT-4.1`), compared to under 1.5% in the zero-shot setting, which is omitted. Representative examples are shown in Figure 2.

While a few recent studies train models to explicitly verbalize their confidence (Band et al., 2024; Kapoor et al., n.d.; Stengel-Eskin et al., 2024; Jang et al., 2024), most rely on complex procedures such as reinforcement learning or classifier probing. Moreover, they report little evidence of generalization in zero-shot settings or under chain-of-thought (CoT) reasoning. At the same time, state-of-the-art LLMs increasingly tackle challenging problems by generating explicit CoT traces (Q. Zhou et al., 2023; B. Chen et al., 2023; X. Wang et al., 2023). The reliability of verbalized confidence within such CoT reasoning, however, remains largely unexplored.

In contrast, this work shows that even a simple confidence-supervised fine-tuning (CSFT) approach, under suitable conditions, can yield well-calibrated verbalized confidence in CoT reasoning. Furthermore, we demonstrate that the model can autonomously adjust its response length and exhibit emergent self-verification behavior as a function of its uncertainty. Specifically, without reasoning supervision, the model learns to modulate its reasoning process while being trained to generate an answer, followed by a verbalized confidence score. As illustrated in Figure 1, low-confidence predictions result in longer outputs with self-check phrases such as “recalculate” or “let me double-check”, whereas high-confidence responses are shorter and more decisive. This phenomenon emerges even without CoT reasoning guidance in training, and is consistently observed across GSM8K, MATH-500, and ARC-Challenge. These findings suggest that verbalized confidence can serve not only as a calibration target but also as an effective training signal that encourages more deliberate chain-of-thought generation, ultimately leading to improved reasoning accuracy.

Our contributions are as follows:

- We propose CSFT, a simple confidence-supervised fine-tuning method using the problems and the corresponding synthetic self-confidence labels that enables reliable verbalized confidence in CoT reasoning tasks.
- We demonstrate that CSFT elicits emergent self-verification behavior without requiring reasoning supervision, manifesting as a systematic relationship between confidence and output length.
- We analyze how prompting style, regularization strength, and reasoning depth affect this phenomenon, and demonstrate generalization to held-out reasoning tasks.

These results provide a scalable path toward building uncertainty-aware LLMs using standard SFT pipelines, without architectural modifications or post-hoc correction. CSFT not only improves calibration but also guides model behavior toward safer and more interpretable reasoning.

2. Related Works

2.1. Confidence Calibration in LLMs

Calibration in LLMs has been studied from various perspectives. Likelihood-based methods estimate uncertainty using token-level entropy, sequence probabilities, or generation variance (Desai and Durrett, 2020; Nguyen et al., 2024; Kadavath et al., 2022). These methods are helpful for model-side diagnostics, but they do not yield human-readable confidence statements. Verbalized confidence, where models explicitly articulate how sure they are, has emerged as a more user-friendly and interpretable alternative. (Band et al., 2024; Stengel-Eskin et al., 2024). However, most existing approaches focus on short-form declarative QA and require manual scalar labels (Lin et al., 2024) or classifier-based tuning (Kapoor et al., n.d.; Jang et al., 2024), without an understanding of the dynamics of reasoning. In the context of CoT reasoning, recent work has reported that instruction-tuned or reasoning-supervised models (A. Yang et al., 2025; Guo et al., 2025) exhibit better calibration under zero-shot inference (Yoon et al., 2025). However, these findings remain observational and do not examine how calibration can be systematically induced or controlled in reasoning tasks. In contrast, our work provides the first direct evidence that fine-tuning with weak, self-derived confidence labels, obtained through consistency across sampled answers, can induce improved calibration and emergent self-verification in reasoning, even in models with no prior exposure to CoT supervision.

2.1.1. Self-Verification and CoT Optimization

Self-verification has emerged as a desirable property for LLMs, with prior work demonstrating that models capable of revisiting and refining their reasoning tend to achieve higher accuracy and robustness, especially on complex tasks (X. Wang et al., 2023). Accordingly, a growing body of research has focused on optimizing the structure of CoT outputs, either by making them more concise (Nayab et al., 2024; Team et al., 2025) or by generating longer and more reflective traces (Guo et al., 2025). However, these approaches typically target a fixed generation style, without conditioning on the model’s internal uncertainty. In reality, effective reasoning should adapt to the model’s confidence: when confidence is low, the model should elaborate and verify; when confidence is high, a brief and decisive answer may suffice. Most existing methods do not capture this dynamic. Reinforcement learning-based approaches train models to favor verifiable traces (B. Chen et al., 2023; Zhao et al., 2025; Shafayat et al., 2025), but require reward shaping and large-scale tuning. We show that a single round of fine-tuning on scalar confidence labels derived from self-consistency induces CoT behaviors that adaptively reflect the model’s uncertainty.

3. Confidence-Supervised Fine-Tuning

We introduce Confidence-Supervised Fine-Tuning (CSFT), a simple yet effective method for calibrating verbalized confidence in LLMs under reasoning scenarios, without requiring explicit supervision of the reasoning process. CSFT fine-tunes the model to calibrate its verbalized confidence, while also producing a CoT reasoning trace and final answer, so that the reported confidence more accurately reflects the model’s belief in the correctness of its answer.

Given a question q, the decoder is trained to generate a structured response consisting of: (i) a CoT reasoning trace r and final answer a, enclosed within <think> ... </think> and <answer> ... </answer> tags, respectively; followed by (ii) a suffix confidence prompt (see Appendix C), which elicits a discrete confidence score $c \in \{0, 10, \ldots, 100\}$ expressed in <confidence> ... </confidence> tags. Only the confidence score c is supervised during training; both the reasoning r and the answer a remain unconstrained.

In this section, we used Low Rank Adaptation (LoRA; Hu et al., 2021) method to fine-tune the LLM model f_{θ_0}. Here, we denote θ_0 as fixed pre-trained parameters and θ as fixed pre-trained parameters with additional learnable LoRA weights.

Self-Confidence label. To compute the confidence label, we first sample K full generations $\{(r^{(i)}, a^{(i)})\}_{i=1}^{K} \sim f_\theta(\cdot \mid q)$ and estimate the empirical accuracy as

$$\hat{p}(q) = \frac{1}{K} \sum_{i=1}^{K} \mathbb{1}\left[a^{(i)} = a^\star\right], \tag{1}$$

where $a^\star$ denotes the gold answer. To determine whether each answer $a^{(i)}$ matches the gold answer $a^\star$, we first parsed the value between <answer> ... </answer> from the LLM response to extract $a^{(i)}$, and then checked whether it exactly matched the gold answer. Then, the self-confidence label was obtained by discretizing the accuracy: $c = \lfloor 100 \cdot \hat{p}(q) \rfloor$.

Training objective. Let T_c denote the token positions corresponding to the entire confidence span, including the <confidence> ... </confidence> tags. CSFT minimizes the masked cross-entropy loss over these positions:

$$\mathcal{L}_{\text{CSFT}} = -\sum_{t \in T_c} \log p_\theta(y_t \mid y_{<t}, q), \tag{2}$$

where $p_\theta(y)$ indicates the predicted probability of token y from LLM f_θ. And optionally, we add a KL regularization term over the CoT and answer spans (including their respective tags), where the corresponding token positions are denoted by T_{KL}, to encourage the model to remain close to the pretrained distribution:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{CSFT}} + \lambda \sum_{t \in T_{\text{KL}}} \text{KL}(p_\theta \parallel p_{\theta_0}), \tag{3}$$

where λ is a weighting hyperparameter, and p_θ and p_{θ_0} represents $p_\theta(\cdot \mid y_{<t}, q)$ and $p_{\theta_0}(\cdot \mid y_{<t}, q)$, respectively. Unless otherwise noted, we set $\lambda = 0$.

Calibration effect. Because the target c reflects the empirical accuracy of the model's own generations, CSFT encourages alignment between predicted confidence and actual correctness. This leads to improved calibration, as measured by standard metrics such as expected calibration error (ECE; Naeini et al., 2015) in CoT reasoning tasks.

4. Experiments

Experimental Setup. We construct our training corpus by sampling $K = 10$ CoT traces and their corresponding answers for each question q from the GSM8K (Cobbe et al., 2021) training split, and assigning a self-consistency label based on the proportion of sampled answers that match the gold answer. Using this signal, we fine-tune `LLaMA3.2-3B-Instruct` (Grattafiori et al., 2024) and `Qwen2.5-1.5B-Instruct` (A. Yang et al., 2025), and evaluate accuracy, Area Under the Receiver Operating Characteristic curve (AUROC), and calibration metrics (ECE, Brier Score) on the GSM8K test set as well as on the held-out reasoning benchmarks MATH-500 (Lightman et al., 2023) and ARC-Challenge (Clark et al., 2018). Further experimental details are provided in Appendix A.

4.1. Main Experiments

Evaluation on GSM8K dataset. Table 1 show results on the GSM8K test set, where our model was trained using CSFT with the GSM8K training dataset. Our method consistently improves all calibration metrics and accuracy over the pre-trained baseline, not only on `LLaMA3.2-3B-Instruct` but also on `Qwen2.5-1.5B-Instruct`. Beyond better alignment between predicted confidence and correctness, we observe that CSFT induces desirable reasoning behaviors, such as internal error checking and more deliberate output construction—hallmarks of self-verification, particularly in low-confidence cases. Refer to Fig. 2 to see an example. And in such low-confidence cases, responses based on self-verification can serve as a mechanism for users to view the predicted model confidence as more reliable.

Unseen CoT Tasks. To evaluate the generalization capability of the model trained with CSFT, we test it on two reasoning benchmarks—MATH-500 and ARC-Challenge—that are structurally, topically, and cognitively distinct from GSM8K. These benchmarks allow us to assess how well the learned reasoning patterns transfer to the unseen domain and more challenging problem distributions. As shown in Table 1, CSFT improves generalization to unseen CoT tasks for both models. On MATH-500, `LLaMA3.2-3B-Instruct` achieves a +37% accuracy gain and 63% ECE reduction, while `Qwen2.5-1.5B-Instruct` yields a +2.5% accuracy gain and 32% lower ECE. On ARC-Challenge, LLaMA improves accuracy by 6.3% and ECE by 71%, and Qwen shows 9.1% higher accuracy and 33% better calibration. These results indicate that our self-consistency-based cali-

Table 1: Calibration results on the in-distribution GSM8K and two held-out reasoning benchmarks. For datasets, ✓: 'seen' during CSFT, ✗: 'unseen' during CSFT. For the metrics, ↓: lower is better and ↑: higher is better.

Dataset	Model	Method	AUROC (↑)	ACC (↑)	ECE (↓)	BS (↓)	Avg. Len.
GSM8K (✓)	`LLaMA3.2-3B-Instruct`	Pre-trained CSFT	50.57 81.25	68.68 71.34	0.2065 0.0568	0.2549 0.1450	226.66 288.71
	`Qwen2.5-1.5B-Instruct`	Pre-trained CSFT	49.59 67.67	67.85 69.63	0.1928 0.0552	0.2915 0.2285	250.21 291.70
MATH-500 (✗)	`LLaMA3.2-3B-Instruct`	Pre-trained CSFT	49.57 62.97	41.20 56.60	0.4730 0.1776	0.4800 0.3059	416.70 559.33
	`Qwen2.5-1.5B-Instruct`	Pre-trained CSFT	59.91 60.27	55.00 56.40	0.3786 0.2590	0.2978 0.2629	444.86 477.96
ARC-Challenge (✗)	`LLaMA3.2-3B-Instruct`	Pre-trained CSFT	53.89 72.58	65.36 69.45	0.2251 0.0647	0.2738 0.1853	210.50 293.13
	`Qwen2.5-1.5B-Instruct`	Pre-trained CSFT	54.08 61.63	52.07 56.82	0.1660 0.1107	0.2680 0.2584	105.55 114.85

bration transfers beyond the training task, improving both confidence alignment and problem-solving ability on diverse reasoning challenges.

4.2. Self-Verification Behaviour

In this section, we analyze how training with CSFT induces self-verification behavior and how this behavior correlates with predicted confidence levels. We further support our analysis with concrete examples. First, Figure 1 presents two key relationships: (1) the average CoT token length as a function of predicted confidence, and (2) the proportion of answers that trigger self-verification across different confidence levels.

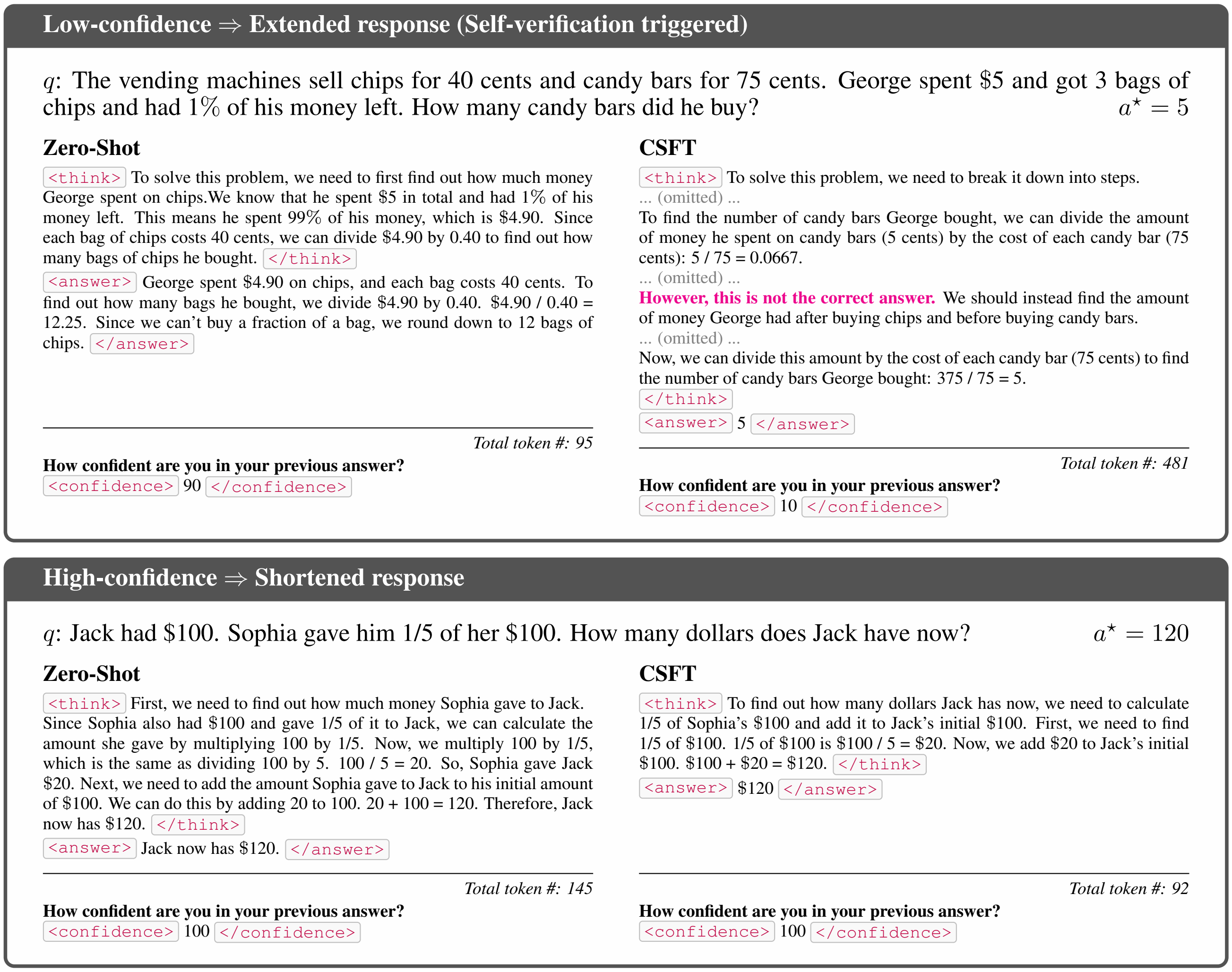
The results in Figure 1 show a clear trend: CSFT-trained models generate significantly longer outputs when their confidence is low. In particular, for the lowest-confidence bin (0), the average output length is nearly five times longer than that of the zero-shot baseline. This indicates that the model compensates for low confidence by engaging in extended reasoning, suggesting that self-verification is an emergent behavior tied to uncertainty. This increase in length is closely accompanied by a high self-verification rate, with nearly all corrected answers involving explicit verification behaviors. Moreover, this self-verification behavior under low-confidence scenarios can serve as a valuable mechanism for end users relying on LLM responses. It provides an implicit signal that the model is uncertain and is actively working to validate its answer, thereby enhancing the perceived reliability of the predicted confidence from the user's perspective. As shown in Appendix B.1, both the length and self-verification patterns generalize beyond the training distribution, emerging similarly on unseen CoT tasks as well.

Qualitative Examples. In Figure 2, we provide qualitative examples comparing responses generated by the pretrained baseline and the model fine-tuned with CSFT. The results highlight that CSFT-trained models dynamically adjust both the length and structure of their reasoning based on the predicted confidence. Low-confidence cases elicit longer, reflective traces with internal correction, whereas high-confidence responses tend to be brief and decisive.

4.3. Ablation Studies

Prompt Position. The *suffix* setting, in which the confidence prompt is appended after the model's answer, is described in § 3. In addition, we evaluate a *prefix* variant where the confidence prompt is inserted immediately after the question and before any reasoning begins (see Appendix C). This placement allows the model to condition its reasoning on anticipated confidence, which may affect both generation and calibration. As shown in Figure 3, the prefix setting consistently yields lower accuracy than the zero-shot baseline, with performance sharply degrading when KL regularization is removed (i.e., $\lambda = 0$). This suggests that without constraints on the CoT and answer spans, the model may overfit to expressing uncertainty rather than reasoning accurately, effectively learning to be confidently wrong or confidently uncertain. In contrast, the suffix setting shows more favorable behavior. When KL regularization is removed, both calibration and accuracy improve compared to the zero-shot baseline. These results highlight a key trade-off: prefix prompting influences the generation process and requires regularization to remain effective, while suffix prompting is more stable because it does not interfere with the model's reasoning.

KL Regularization. Figure 3 We investigate the effect of varying the KL regularization weight λ on model per-



Low-confidence ⇒ Extended response (Self-verification triggered)

q: The vending machines sell chips for 40 cents and candy bars for 75 cents. George spent $5 and got 3 bags of chips and had 1% of his money left. How many candy bars did he buy? $a^\star = 5$

Zero-Shot

<think> To solve this problem, we need to first find out how much money George spent on chips.We know that he spent $5 in total and had 1% of his money left. This means he spent 99% of his money, which is $4.90. Since each bag of chips costs 40 cents, we can divide $4.90 by 0.40 to find out how many bags of chips he bought. </think>

<answer> George spent $4.90 on chips, and each bag costs 40 cents. To find out how many bags he bought, we divide $4.90 by 0.40. $4.90 / 0.40 = 12.25. Since we can't buy a fraction of a bag, we round down to 12 bags of chips. </answer>

Total token #: 95

How confident are you in your previous answer?

<confidence> 90 </confidence>

CSFT

<think> To solve this problem, we need to break it down into steps.

... (omitted) ...

To find the number of candy bars George bought, we can divide the amount of money he spent on candy bars (5 cents) by the cost of each candy bar (75 cents): 5 / 75 = 0.0667.

... (omitted) ...

However, this is not the correct answer. We should instead find the amount of money George had after buying chips and before buying candy bars.

... (omitted) ...

Now, we can divide this amount by the cost of each candy bar (75 cents) to find the number of candy bars George bought: 375 / 75 = 5.

</think>

<answer> 5 </answer>

Total token #: 481

How confident are you in your previous answer?

<confidence> 10 </confidence>

High-confidence ⇒ Shortened response

q: Jack had $100. Sophia gave him 1/5 of her $100. How many dollars does Jack have now? $a^\star = 120$

Zero-Shot

<think> First, we need to find out how much money Sophia gave to Jack. Since Sophia also had $100 and gave 1/5 of it to Jack, we can calculate the amount she gave by multiplying 100 by 1/5. Now, we multiply 100 by 1/5, which is the same as dividing 100 by 5. 100 / 5 = 20. So, Sophia gave Jack $20. Next, we need to add the amount Sophia gave to Jack to his initial amount of $100. We can do this by adding 20 to 100. 20 + 100 = 120. Therefore, Jack now has $120. </think>

<answer> Jack now has $120. </answer>

Total token #: 145

How confident are you in your previous answer?

<confidence> 100 </confidence>

CSFT

<think> To find out how many dollars Jack has now, we need to calculate 1/5 of Sophia's $100 and add it to Jack's initial $100. First, we need to find 1/5 of $100. 1/5 of $100 is $100 / 5 = $20. Now, we add $20 to Jack's initial $100. $100 + $20 = $120. </think>

<answer> $120 </answer>

Total token #: 92

How confident are you in your previous answer?

<confidence> 100 </confidence>

Figure 2: **Confidence reshapes reasoning behavior.** Left: In the low-confidence case (top), the CSFT model generates a long reasoning trace with explicit self-verification, eventually arriving at the correct answer, while the zero-shot model fails. Right: In the high-confidence case (bottom), both models give the correct answer, but the CSFT response is significantly more concise, reflecting confidence-aware brevity.

formance, focusing on its impact near $\lambda = 0$ (see Figure 3, Zoom-in figure). The motivation for this analysis is to examine whether performance gains at low λ are stable or merely an artifact of tuning. For the prefix setting, we observe that performance rapidly deteriorates as KL regularization is removed. This suggests that, without a constraint to preserve the pretrained distribution over CoT and answer spans, the model may exploit the freedom to optimize ECE at the expense of actual reasoning quality. In effect, the model becomes well-calibrated but confidently incorrect. In contrast, the suffix setting remains stable or even improves in the absence of KL regularization. Since confidence is predicted independently after the full generation, removing the KL constraint does not impair reasoning quality, and may in fact allow for better post hoc alignment of confidence with correctness. These results highlight the importance of controlling model behavior when confidence supervision is introduced at generation time (prefix), as opposed to after-the-fact (suffix).

Table 2: Ablation analysis on the impact of confidence label quality and the inclusion of confidence prompt. Results are reported as differences relative to CSFT on GSM8K using `LLaMA3.2-3B-Instruct`.

Variant	ACC	ECE	Avg. Len.
w/o Correct label	-2.14	+0.05	-49.36
w/o Conf question	Training collapsed		

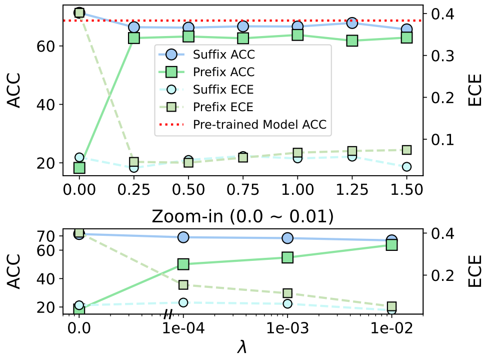


Figure 3: Test accuracy and ECE on GSM8K using CSFT-trained `LLaMA3.2-3B-Instruct`, evaluated under varying KL weights. Prefix performance declines without KL, whereas suffix remains stable or improves.

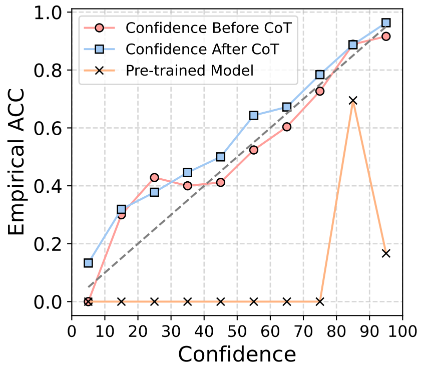


Figure 4: Reliability curves of CSFT-trained `LLaMA3.2-3B-Instruct` on GSM8K, comparing predicted confidence when elicited before vs. after CoT generation. The broadly similar calibration profiles suggest that the model's confidence reflects internal uncertainty rather than simply CoT length.

Label Quality and Confidence Question. To evaluate the role of confidence supervision and the design of the confidence prompt in enabling stable training and achieving strong performance, we conduct two ablation studies, as summarized in Table 2. First, we randomly assign confidence labels within the `<confidence>` tag, breaking the link between prediction quality and label supervision. Second, we remove the explicit confidence prompt, instead asking the model to generate a scalar confidence directly after the final answer using a repeated `<answer>` tag. In the first setting, performance significantly degrades across all metrics, confirming that accurate supervision is critical for learning calibrated confidence. In the second setting, the model fails to train altogether, suggesting that without an explicit instruction to predict confidence, the model cannot ground the meaning of the scalar and collapses.

Impact of CoT Visibility on the Confidence. To test whether the model's confidence relies on observing the length or content of the generated CoT, we compare calibration when confidence is elicited before versus after CoT generation. As shown in Figure 4, the two reliability curves are broadly similar, suggesting that the model does not depend heavily on CoT visibility and instead bases its confidence on internal uncertainty.

4.4. Confidence-Guided Reasoning Path Refinement

Table 3: Manual rethinking improves accuracy in low-confidence bins on GSM8K test set using `LLaMA3.2-3B-Instruct` (**ΔACC**).

Bin	**0**	**10**	**20**	**30**
ΔACC	+0.5625	+0.5524	+0.3158	+0.2069
Count	16	143	38	29

When a model is well-calibrated, its verbalized confidence serves as a trustworthy signal for downstream decision-making and control. As demonstrated previously, the CSFT-trained model is capable of accurately predicting its confidence even before generating the full CoT. This capability opens up the possibility of guiding the reasoning trajectory from the very beginning.

We exploit this by manually redirecting low-confidence samples toward alternative reasoning paths. Specifically, if the model expresses low confidence in its initial output(elicited via the prefix prompt shown in Figure 9), we initiate a new reasoning attempt with an altered or more structured prompt (as shown in Figure 14), without waiting for failure. This preemptive rethinking mechanism enables selective refinement with minimal additional cost.

As shown in Table 3, this confidence-aware rethinking strategy substantially improves accuracy in the low confidence bins. For example, in the 0–10 confidence range, accuracy improves by over 55 percentage points. This result underscores the utility of confidence not just for post hoc calibration, but also for guiding efficient and cost-aware reasoning-time improvement.

5. Conclusion and Future Work

To develop well-calibrated LLMs capable of expressing trustworthy verbalized confidence in CoT reasoning, we propose a simple yet effective fine-tuning method called CSFT. For training, we construct a synthetic dataset based on the GSM8K training set, where each problem is paired with a self-generated confidence label.

Empirically, we demonstrate that LLMs fine-tuned with

CSFT achieve substantial improvements across multiple evaluation metrics—including accuracy and ECE—on both the GSM8K test split and out-of-domain benchmarks. Remarkably, we also observe that CSFT elicits the emergence of self-verification behavior, particularly in low-confidence scenarios, despite not providing any explicit supervision related to reasoning strategies during training. Moreover, this self-verification behavior in low-confidence scenarios can act as a valuable mechanism for end users interacting with LLMs. It serves as an implicit indicator that the model recognizes its own uncertainty and is actively attempting to validate its response.

Looking forward, several directions emerge. First, since we observe that confidence can be elicited *prior* to reasoning, it may be possible to predict the downstream cost of a reasoning trajectory (e.g., output length or compute usage) from the initial confidence. This opens up opportunities for confidence-conditioned inference policies that balance accuracy and efficiency. Second, while self-verification is desirable under uncertainty, we find that some low-confidence generations enter excessively long or redundant reasoning loops, potentially reflecting local minima in the generation dynamics. Third, one could explore confidence-aware *steering* of CoT trajectories, or use latent confidence signals to trigger rethink-style interventions without incurring full-generation overhead.

References

Achiam, Josh et al. (2023). “Gpt-4 technical report”. In: *arXiv preprint arXiv:2303.08774* (cit. on p. 1).

Band, Edward et al. (2024). “Linguistic Calibration of Long-Form Generations”. In: *Advances in Neural Information Processing Systems*. URL: https://proceedings.neurips.cc/paper_files/paper/2024/file/4b8eaf3bcdc105423a972ed90eb07217-Paper-Conference.pdf (cit. on pp. 1, 2).

Brier, Glenn W (1950). “Verification of forecasts expressed in terms of probability”. In: *Monthly weather review* 78.1, pp. 1–3 (cit. on p. 9).

Chen, Bailin et al. (2023). “Program-of-Thoughts Prompting: Disentangling Computation from Reasoning for Numerical Reasoning Tasks”. In: *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 9757–9778 (cit. on pp. 1, 2).

Clark, Peter et al. (2018). “Think you have solved question answering? try arc, the ai2 reasoning challenge”. In: *arXiv preprint arXiv:1803.05457* (cit. on pp. 3, 9).

Cobbe, Karl et al. (2021). “Training verifiers to solve math word problems”. In: *arXiv preprint arXiv:2110.14168* (cit. on pp. 3, 9).

Desai, Shrey and Greg Durrett (2020). “Calibration of Pre-trained Transformers”. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*. URL: https://aclanthology.org/2020.emnlp-main.763/ (cit. on p. 2).

Du, Xuefeng, Chaowei Xiao, and Sharon Li (2024). “Haloscope: Harnessing unlabeled llm generations for hallucination detection”. In: *Advances in Neural Information Processing Systems* 37, pp. 102948–102972 (cit. on p. 1).

Goh, Ethan et al. (2024). “Large language model influence on diagnostic reasoning: a randomized clinical trial”. In: *JAMA Network Open* 7.10, e2440969–e2440969 (cit. on p. 1).

Grattafiori, Aaron et al. (2024). “The llama 3 herd of models”. In: *arXiv preprint arXiv:2407.21783* (cit. on p. 3).

Guo, Daya et al. (2025). “Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning”. In: *arXiv preprint arXiv:2501.12948* (cit. on pp. 1, 2).

Hu, Edward J et al. (2021). “LoRA: Low-Rank Adaptation of Large Language Models”. In: *International Conference on Learning Representations* (cit. on p. 3).

Jang, Chaeyun et al. (2024). “Calibrated Decision-Making through LLM-Assisted Retrieval”. In: *arXiv preprint arXiv:2411.08891* (cit. on pp. 1, 2).

Kadavath, Saurav et al. (2022). “Language Models (Mostly) Know What They Know”. In: *arXiv preprint arXiv:2207.05221*. URL: https://arxiv.org/abs/2207.05221 (cit. on p. 2).

Kapoor, Sanyam et al. (n.d.). “Large Language Models Must Be Taught to Know What They Don’t Know”. In: *The Thirty-eighth Annual Conference on Neural Information Processing Systems* (cit. on pp. 1, 2).

Lightman, Hunter et al. (2023). “Let’s verify step by step”. In: *The Twelfth International Conference on Learning Representations* (cit. on pp. 3, 9).

Lin, Xiang et al. (2024). “Calibrating the Confidence of Large Language Models by Eliciting Self-Reflective Responses”. In: *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*. URL: https://aclanthology.org/2024.emnlp-main.173.pdf (cit. on p. 2).

Naeini, Mahdi Pakdaman, Gregory Cooper, and Milos Hauskrecht (2015). “Obtaining well calibrated probabilities using bayesian binning”. In: *Association for the Advancement of Artificial Intelligence (AAAI)* (cit. on pp. 3, 9).

Nayab, Sania et al. (2024). “Concise thoughts: Impact of output length on llm reasoning and cost”. In: *arXiv preprint arXiv:2407.19825* (cit. on p. 2).

Nguyen, Hy et al. (2024). “Semantic Entropy Probes: Robust and Cheap Hallucination Detection in Large Language Models”. In: *arXiv preprint arXiv:2406.15927*. URL: https://arxiv.org/abs/2406.15927 (cit. on p. 2).

Qiu, Jianing et al. (2024). “LLM-based agentic systems in medicine and healthcare”. In: *Nature Machine Intelligence* 6.12, pp. 1418–1420 (cit. on p. 1).

Shafayat, Sheikh et al. (2025). “Can Large Reasoning Models Self-Train?” In: *arXiv preprint arXiv:2505.21444* (cit. on p. 2).

Stengel-Eskin, Elias, Peter Hase, and Mohit Bansal (2024). “LACIE: Listener-aware finetuning for calibration in large language models”. In: *Advances in Neural Information Processing Systems* 37, pp. 43080–43106 (cit. on pp. 1, 2).

Takayanagi, Takehiro et al. (2025). “Are Generative AI Agents Effective Personalized Financial Advisors?” In: *arXiv preprint arXiv:2504.05862* (cit. on p. 1).

Team, Kimi et al. (2025). “Kimi k1. 5: Scaling reinforcement learning with llms”. In: *arXiv preprint arXiv:2501.12599* (cit. on p. 2).

Wang, Xuezhi et al. (2023). “Self-Consistency Improves Chain-of-Thought Reasoning in Language Models”. In: *Proceedings of the 40th International Conference on Machine Learning (ICML)*, pp. 24565–24585 (cit. on pp. 1, 2).

Yang, An et al. (2025). “Qwen3 technical report”. In: *arXiv preprint arXiv:2505.09388* (cit. on pp. 2, 3).

Yoon, Dongkeun et al. (2025). “Reasoning Models Better Express Their Confidence”. In: *arXiv preprint arXiv:2505.14489* (cit. on p. 2).

Zhao, Xuandong et al. (2025). “Learning to Reason without External Rewards”. In: *arXiv preprint arXiv:2505.19590* (cit. on p. 2).

Zhou, Kaitlyn et al. (2024). “Relying on the Unreliable: The Impact of Language Models’ Reluctance to Express Uncertainty”. In: *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3623–3643 (cit. on p. 1).

Zhou, Qinyuan et al. (2023). “Least-to-Most Prompting Enables Complex Reasoning in Large Language Models”. In: *Advances in Neural Information Processing Systems 36 (NeurIPS)* (cit. on p. 1).

A. Experimental details

In this section, we provide detailed information on the models and datasets used in our experiments, along with formal definitions of the calibration metrics employed for evaluation. Specifically, we describe the two instruction-tuned language models used: `LLaMA3.2-3B-Instruct` and `Qwen2.5-1.5B-Instruct`. For datasets, we include:

- **GSM8K** (Cobbe et al., 2021); a dataset of 7.47k grade-school math word problems designed to test step-by-step reasoning. We use 10% and 20% of the original training set (0.75k and 1.49k examples, respectively) for training and validation, and the full test set of 1.32k examples for evaluation. Available at huggingface.co/openai/gsm8k.
- **MATH-500** (Lightman et al., 2023); a subset of the MATH dataset consisting of 500 diverse high school level problems covering algebra, geometry, calculus, and more. Used solely for evaluation. Available at huggingface.co/HuggingFaceH4/MATH-500.
- **ARC-Challenge** (Clark et al., 2018); a multiple-choice science and commonsense QA benchmark containing 1.17k test questions that require reasoning beyond surface-level cues. We use the test set for evaluation. Available at huggingface.co/allenai/ai2_arc.

A.1. Model and datasets

Table 4: Training hyperparameters used for CSFT fine-tuning across both models.

Hyperparameter	**Value**
Batch size	1
Gradient accumulation	16
Learning rate	[1e-5, 1e-04]
Optimizer	AdamW
Weight decay	0.0
Warmup ratio	0.0
Max sequence length	1024
KL regularization (λ)	0.0
Training steps	2500
Checkpoint selection	Best dev loss
LoRA configuration	
LoRA rank (r)	128
LoRA alpha	32
LoRA dropout	0.1
LoRA target modules	q_{proj}, v_{proj}

A.1.1. Calibration metrics

- **Expected Calibration Error** (ECE; Naeini et al., 2015):

$$\text{ECE} = \sum_{m=1}^{M} \frac{|B_m|}{n} |\text{acc}(B_m) - \text{conf}(B_m)|$$

where B_m is the set of predictions in bin m, acc(B_m) is the accuracy, and conf(B_m) is the average confidence of the predictions in that bin. ECE measures how well the model's predicted probabilities are calibrated.

- **Brier Score** (BS; Brier, 1950):

$$\text{BS} = \frac{1}{N} \sum_{i=1}^{N} (f_i - y_i)^2$$

where f_i is the predicted probability and y_i is the true label. BS combines both the accuracy and confidence of the predictions, penalizing overconfident and underconfident predictions.

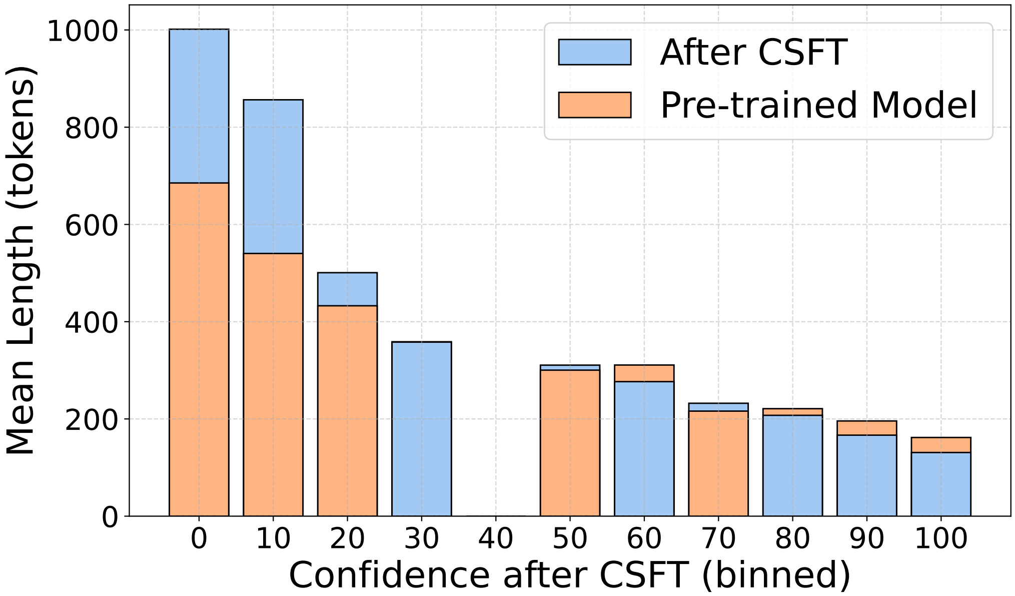


Figure 5: Output length across confidence bins on Math-500, using `LLaMA3.2-3B-Instruct` fine-tuned with CSFT. The model generates significantly longer responses when confidence is low, while high-confidence predictions tend to be more concise.

B. Additional Results

B.1. Length Analysis on Held-out CoT Tasks

Figure 5 and Figure 6 present an analysis of model outputs on Math-500 and ARC-Challenge—two held-out CoT tasks not seen during CSFT training. As shown in both figures, output length increases in low-confidence bins. In the case of Math-500, there is also a clear trend toward more concise responses in high-confidence bins. These results demonstrate that the length modulation effect reported in the main paper is not restricted to the training distribution but generalizes to unseen tasks. In other words, CSFT enables the model to internalize the ability to adjust response length based on its uncertainty, suggesting a deeper transformation in its reasoning behavior.

C. Prompt Examples

This section presents the prompt templates used in our experiments for eliciting model reasoning, answers, and confidence scores. We include both the *prefix* prompt, where the confidence is generated before reasoning begins, and the *suffix* prompt, where confidence is predicted after the final answer. All prompts follow a standardized format to ensure consistent supervision during CSFT and reliable evaluation during inference.

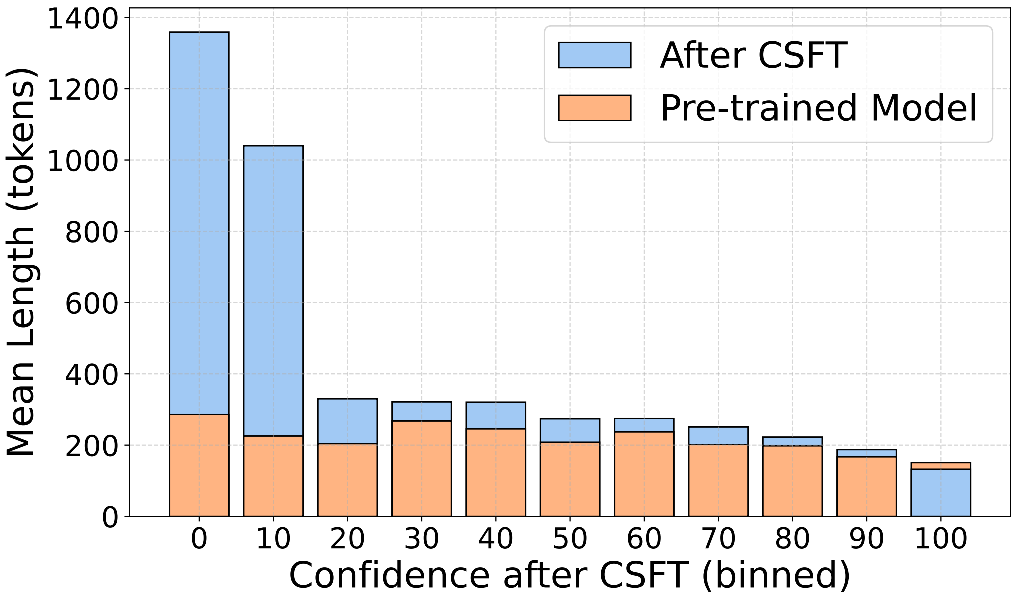


Figure 6: Output length across confidence bins on ARC-Challenge, using `LLaMA3.2-3B-Instruct` fine-tuned with CSFT. The same trend of confidence-conditioned length modulation emerges, despite the task being unseen during training.

»> Base Reasoning Prompt

This is a conversation between **User** and **Assistant**.
The User asks a question, and the Assistant provides a solution.
Before answering, the Assistant reasons through the problem step-by-step.
The reasoning is enclosed within `<think> ... </think>`, and the final answer within `<answer> ... </answer>`.

Example:
`{question}`
`<think>{step-by-step reasoning}</think>`
`<answer>{final answer}</answer>`

Now, respond to the following using the **exact same format**:
`<question>`

Figure 7: Base prompt format used to elicit step-by-step reasoning and final answer generation.

»> Suffix Confidence Prompt

Please respond with a score from 0 to 100 in `<confidence> </confidence>` tags.
How confident are you in your previous answer?

Figure 8: Suffix confidence prompt used to elicit model confidence *after* the full reasoning and answer generation. Since confidence is predicted post hoc, this setup allows for calibration without interfering with the model's original reasoning or final answer.

»> Prefix Confidence Prompt

This is a conversation between **User** and **Assistant**.
The User asks a question, and the Assistant provides a thoughtful, reasoned answer.
Before answering, the Assistant first reasons through the problem step-by-step.
The reasoning is enclosed in `<think> ... </think>` tags.
The final answer is enclosed in `<answer> ... </answer>` tags.
A confidence score is then provided in `<confidence> ... </confidence>` tags, representing the Assistant's certainty as a **continuous value between 0 and 100**.

Example:

```
{question}
<think>{step-by-step reasoning}</think>
<answer>{final answer}</answer>
<confidence>{confidence}</confidence>
```

Now, answer the following in **exactly** the same format:
`<question>`

Figure 9: Prefix confidence prompt used to elicit model confidence *before* reasoning begins. By conditioning the generation on anticipated confidence, this prompt not only guides the model's uncertainty expression but also influences the reasoning path and final answer.

»> GSM8K Parsing Prompt

Instruction:
We have a user's question and a model's generated response:
Your task:
1. Carefully read the question and the generated response in **Example 6 only**.
2. Extract the final answer based on the following rules:

- If the response contains a number (with or without units), **extract only the numeric value**.
- If the response is purely textual (no numbers), **extract the exact string as it appears**.

3. Use the following output format: `Model's Final Answer is: [Your extracted answer]`

Rules:

- Only process **Example 6** for extraction. Ignore all other examples.
- Do not include units, symbols, or extra text when extracting numbers.
- Provide the answer strictly in the requested format without additional explanations.

Examples
Example 1: Model's Generated Response: It takes about 160 minutes.
`Model's Final Answer is: 160`
Example 2: Model's Generated Response: The nearest star is approximately 4.24 light years away.
`Model's Final Answer is: 4.24`
Example 3: Model's Generated Response: The tallest mountain is Mount Everest.
`Model's Final Answer is: Mount Everest`
Example 4: Model's Generated Response: It weighs 5 kg.
`Model's Final Answer is: 5`
Example 5: Model's Generated Response: 81 + 221 - 24 = 278.
`Model's Final Answer is: 278`

Example 6: Model's Generated Response: `<answer_text>`

Figure 10: Prompt used to extract final answers from model-generated responses on GSM8K.

»> Math500 Matching Prompt

Instruction:
You are given the true answer and the final answer generated by a model for a math problem.
Your task:

1. Only examine **Example 6**.
2. Compare the **model's final answer** and the **true answer**.
3. Respond with `"yes"` if they exactly match, otherwise respond with `"no"`.
4. Do not include any explanation or extra words—just respond with `"yes"` or `"no"`.

Examples
Example 1:
True Answer: 0.5
Model Answer: 1/2
Is it correct?: `yes`
Example 2:
True Answer: 24
Model Answer: 22
Is it correct?: `no`
Example 3:
True Answer: 8
Model Answer: 32 / 4 = 8
Is it correct?: `yes`
Example 4:
True Answer: `\frac{10}{4}`
Model Answer: `\frac{9}{4}`
Is it correct?: `no`
Example 5:
True Answer: 3
Model Answer: `\frac{15}{5}`
Is it correct?: `yes`

Example 6:
True Answer: `<true_answer>`
Model Answer: `<model_answer>`
Is it correct?:

Figure 11: Matching prompt for evaluating exact answer agreement on Math500. Designed to assess correctness by comparing model output with the ground truth in a strict yet interpretable format.

»> ARC-Challenge Parsing Prompt

Instruction:
A user's question provides four choices formatted exactly as:

```
A. <option A>
B. <option B>
C. <option C>
D. <option D>
```

We also have the model's generated response.
Your task:

1. Read **Example 6 only**.
2. Decide which single choice (`A`, `B`, `C`, or `D`) the model ultimately selected, following these rules:
   - **Letter match** – If the response explicitly includes the letter 'A', 'B', 'C', or 'D' (optionally followed by punctuation), extract **only that letter**.
   - **Text match** – If no letter is given, compare the response text (case-insensitive, ignoring punctuation and surrounding spaces) with each option; if it matches exactly one, return the corresponding letter.
   - If both a letter and option text appear, treat the letter as authoritative.
3. Output format (strict):
   `Model's Final Answer is:  <A | B | C | D>`

Do not add explanations or any extra text.

Examples
Example 1:
Choices: A. Paris B. Berlin C. Madrid D. Rome
Model's Generated Response: A. Paris is the capital of France.
`Model's Final Answer is:  A`
Example 2:
Choices: A. 3 B. 4 C. 5 D. 6
Model's Generated Response: The correct option is B.
`Model's Final Answer is:  B`
Example 3:
Choices: A. Spring B. Summer C. Autumn D. Winter
Model's Generated Response: It's usually coldest in winter.
`Model's Final Answer is:  D`

Example 6:
Choices: `{choices}`
Model's Generated Response: `{answer_text}`

Figure 12: Parsing prompt for multiple-choice answer extraction on ARC-Challenge. The rules prioritize explicit letter selection, with fallback to semantic string matching.

»> Pre-CoT Confidence Prompt

Before generating your answer, can you first assess your internal confidence (0–100) in its correctness and state it using '<confidence> </confidence>' tags, then proceed to provide your full answer?

Figure 13: Prompt format for eliciting a model's self-assessed confidence **prior to** generating CoT response.

»> Low Confidence Rethinking Prompt

Your confidence score is low. Rather than following your current reasoning path, pause and explore an alternative approach that is likely to raise your confidence. Think step-by-step and provide a revised answer.

Figure 14: Prompt used when the model reports low confidence, encouraging it to pause and reconsider its reasoning path to generate a more confident response.