# Efficient Test-Time Scaling via Self-Calibration

**Anonymous Author(s)**
Affiliation
Address
email

## Abstract

While increasing test-time computation with methods like Best-of-N sampling and Self-Consistency enhances the quality of Large Language Model (LLM) responses, their fixed sampling strategy is inefficient. These approaches waste computation on simple questions while insufficiently exploring complex ones. We argue that model **confidence** is key to improving this efficiency. However, LLMs are notoriously overconfident, making their confidence estimates unreliable. To address this, we introduce **Self-Calibration**, a method that distills reliable confidence scores from Self-Consistency back into the model itself, enabling accurate confidence estimation in a single forward pass. Building on this, we designed confidence-based efficient test-time scaling methods, such as incorporating an Early-Stopping mechanism for Best-of-N. Experiments across six datasets demonstrate our approach's effectiveness. Notably, on MathQA, our method improved accuracy from **81.0% to 83.6%** with a 16-response budget, confirming the value of our strategy.

## 1 Introduction

While repeated sampling methods like Best-of-N and Self-Consistency can enhance the quality of large language model (LLM) responses [1, 2], their strategy of using a fixed number of samples per query is computationally inefficient and wastes significant resources on simple problems [3, 4]. To address this, existing adaptive sampling strategies [5, 6, 7] often rely on hand-crafted heuristic rules that limit their generalizability. Therefore, it is critical to develop a robust, task- and model-agnostic approach that does not depend on such heuristics.

We propose an efficient test-time scaling method by using model confidence for dynamically sampling adjustment, since confidence can be seen as an intrinsic measure that directly reflects model uncertainty on different tasks. However, extracting accurate confidence can be challenging since LLMs are known to be overconfident in their own responses [8, 9, 10], and their confidences often exceed the actual accuracy, especially in small models. Self-Consistency [11] can provide a relatively accurate confidence estimation by aggregating answer counts from multiple sampled solutions [12], but it again requires sampling numbers of responses for each query.

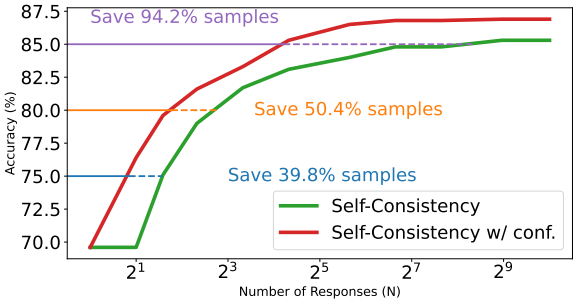



Figure 1: Acc. over response numbers of standard Self-Consistency (SC) vs. confidence-weighted Self-Consistency (SC w/ conf.) on MathQA using our trained Llama-3.1-8B-Instruct model.

To address this, we introduce **Self-Calibration** to train LLMs for accurate confidence estimation in only one forward pass, without requiring any human-labeled data. Specifically, we improve model

Submitted to NeurIPS 2025 Workshop on Efficient Reasoning. Do not distribute.

calibration by distilling Self-Consistency-derived confidence into the model itself. This is done by constructing pseudo training tuples of query, answer, and confidence on a diverse training set. At test time, we design **efficient test-time scaling strategies using these calibrated confidence scores**, such as early stopping for Best-of-N when sampled responses reach a target confidence.

Our confidence-based methods consistently outperform standard approaches with the same sampling budget. For example, our techniques improve MathQA accuracy from 81.0 to 83.6 using an average of just 16 samples. Crucially, our methods also achieve similar performance while using far fewer computational resources. For instance, confidence-weighted Self-Consistency saves 94.2% of the required samples to reach 85.0 accuracy compared to the standard method, proving that reliable confidence scores significantly boost computational efficiency.

# 2 Self-Calibration

In this section, we provide an overview of our proposed Self-Calibration framework, illustrated in Fig. 2. First, we synthesize a set of input-output-confidence tuples $(x, y, c)_i$ from a seed dataset for training, without requiring any ground-truth answer (Sec. 2.2). Using this synthetic dataset, we can train a language model with a combined loss to output calibrated confidence scores (Sec. 2.3).

## 2.1 Confidence Score Estimation

A naive method to obtain a confidence score from an LLM is to calculate $\mathrm{P(True)}$ [13]. This is done by appending a question like, "Is the answer correct? (Yes/No)" to the input-output pair and measuring the probability of the token **Yes**. While this method is computationally cheap due to the KV-cache mechanism [14], empirical results suggest that the resulting $\mathrm{P(True)}$ score is often poorly calibrated, leading the model to be overconfident in its incorrect answers [15]. The proposed solution is to use supervised training to improve this calibration, helping the LLM produce more reliable confidence scores.

## 2.2 Training Data Generation

Our goal is to create a labeled dataset $D_t = (x, y, c)_i$ without human annotations, where $(x, y)$ is a query–response pair and $c$ is an accurate confidence. To achieve this, we first generate multiple candidate answers for each query and ensure diversity via Dynamic Temperature sampling. Next, we calibrate the confidence of each candidate through Soft Self-Consistency, which integrates the model's intrinsic probability estimate with the overall agreement among different responses.

**Soft Self-Consistency Score.** Previous work has shown that self-consistency scores provide strong zero-shot calibration [11], outperforming $\mathrm{P(True)}$ or raw logits as confidence measures [16]. To further enhance the reliability of the confidence score in the training set, we introduce a soft self-consistency score, which integrates $\mathrm{P(True)}$ with self-consistency and offers a more accurate and robust confidence estimation. For each query $x$, we use the LLM to generate $N$ different responses, each with an associated confidence score. Given the set of triplets $(x, y_n, c_n)$ where $1 \leq n \leq N$, we compute the soft self-consistency (SSC) score as:

$$\mathrm{SSC}(y) = \frac{\sum_{i:y_i=y} c_i}{\sum_{i=1}^{N} c_i}.$$

Using this score, we construct the final training set as $(x, y_i, \mathrm{SSC}(y_i))$, where $\mathrm{SSC}(y_i)$ provides a calibrated confidence estimation for each response. For the Llama-3.1-8B-Instruct model on the GSM8K dataset, our SSC method achieves the lowest Expected Calibration Error [17] (ECE, see App. I). SSC scored 3.42, outperforming both P(True) at 12.03 and standard self-consistency at 4.48.

## 2.3 Training Objective

We optimize the model's confidence estimation by minimizing the difference between the predicted confidence and the target confidence using the SmoothL1 loss. To ensure that training on confidence estimation does not degrade the model's reasoning ability, we incorporate the standard generation loss of Chain-of-Thought answers into the objective [18]. Specifically, only responses with confidence scores above a threshold $\eta$ are selected for training to guarantee the quality of the reasoning path. A

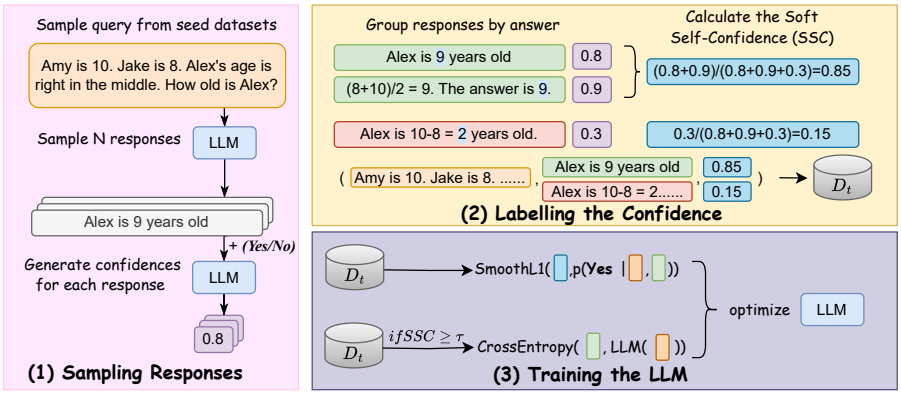



Figure 2: Illustration of the Self-Calibration framework. Given a query from the seed dataset, we sample $N$ responses from the LLM. We use a confidence querying prompt to let LLM assign a confidence score to each response. Responses are then grouped by their answers, and the Soft Self-Consistency (SSC) score is computed for each group. During training, all data tuples contribute to improving the model's calibration, while higher-confidence data is used to enhance the LLM's generation ability.

weighting coefficient $w$ is introduced to balance these two loss terms. The overall loss function is formulated as:

$$\mathcal{L}_{\text{total}}(\theta) = \sum_{(x_j,\, y_j)\in\mathcal{D}} \text{SmoothL1}\Big(p_\theta(\text{Yes} \mid x_j, y_j, I),\ c_j\Big) + \omega \sum_{\substack{(x_i,\, y_i) \\ c_i > \eta}} \Big(-\log\, p_\theta\big(y_i \ \mid\ x_i\big)\Big).$$

# 3 Efficient Test-Time Scaling

In this section, we introduce how to leverage reliable confidence scores from a Self-Calibration trained model to enhance existing test-time scaling methods. The confidence scores serve as a quality indicator for each response, allowing the model to prioritize more reliable answers. We present confidence-guided variants of three popular test-time scaling approaches (Appendix A): Best-of-N, Self-Consistency, and Adaptive Self-Consistency.

## 3.1 Early Stopping with Confidence

Early Stopping improves the efficiency of Best-of-N by terminating the sampling process once a response with sufficient confidence is found. Given a sequential sampling process where each response $y_i$ is assigned a confidence score $c_i$. This means that we continue sampling responses one by one until a response meets the confidence threshold $\tau$, and such a response is selected as the final answer, avoiding unnecessary additional sampling and reducing computational overhead.

## 3.2 Self-Consistency with Confidence

Self-Consistency with Confidence(SC w/ Conf.) extends the traditional Self-Consistency approach by incorporating confidence scores into the voting process. Instead of treating all sampled responses equally, we assign each response $y_i$ a confidence score $c_i$, leading to a weighted aggregation:

$$y \ = \ \arg\max_{z} \sum_{i=1}^{N} c_i\, \mathbb{1}(\, y_i = z\,).$$

This modification ensures that responses with higher confidence contribute more significantly to the final selection, enhancing robustness by prioritizing more reliable predictions.

## 3.3 Adaptive Self-Consistency with Confidence

Similar to Self-Consistency with Confidence, we use confidence as the weight when calculating the relative frequency in Adaptive Self-Consistency. Then Adaptive Self-Consistency with Confidence(ASC w/ Conf.) will follow the rest part in ASC.

$$v_k(z) = \sum_{i=1}^{k} c_i \mathbb{1}(y_i = z), \quad \hat{r}_k(z) = \frac{v_k(z)}{\sum_{i=1}^{k} c_i}.$$

| | Llama-3.1-8B-Instruct | | | Qwen2.5-7B-Instruct | | | DeepSeek-R1-Distill-1.5B | | |
|---|---|---|---|---|---|---|---|---|---|
| Methods | Obj_C. | MathQA | ARC_C. | Obj_C. | MathQA | ARC_C. | Obj_C. | MathQA | ARC_C. |
| Pass@1 | 67.6 | 71.5 | 82.8 | 76.8 | 82.9 | 88.5 | 61.2 | 89.9 | 58.2 |
| SC | 76.0 | 81.0 | 87.1 | 81.2 | 86.3 | **91.2** | **70.8** | 91.6 | 65.6 |
| SC w/ Conf. | **76.8** (+0.8) | **83.6** (+2.6) | **87.7** (+0.6) | 81.2 (0.0) | **87.8** (+1.5) | 90.8 (-0.4) | **70.8** (0.0) | **91.8** (+0.2) | **66.5** (+0.9) |
| Best-of-N | 69.2 | 81.0 | 86.4 | 76.8 | 86.8 | 90.2 | 54.0 | 90.0 | 58.9 |
| Early Stopping | **76.8** (+7.6) | **83.6** (+2.6) | **87.7** (+1.3) | 81.2 (+4.4) | **87.8** (+1.0) | 90.8 (+0.6) | **70.8** (+16.8) | 91.6 (+1.6) | **66.5** (+7.6) |
| ASC | 74.8 | 80.0 | 86.5 | **81.6** | 86.2 | 90.6 | 70.4 | 91.6 | 64.3 |
| ASC w/ Conf. | 75.2 (+0.4) | 81.9 (+1.9) | 86.6 (+0.1) | **81.6** (0.0) | 87.2 (+1.0) | **91.2** (+0.6) | 70.4 (0.0) | **91.8** (+0.2) | 65.1 (+0.8) |
| ESC | 76.0 | 81.0 | 87.1 | 81.2 | 86.3 | 91.0 | **70.8** | 91.3 | 65.6 |
| RASC | 76.0 | 81.4 | 87.3 | 81.2 | 86.4 | 90.3 | **70.8** | 91.4 | 65.8 |

Table 1: Accuracy comparison of different test-time scaling methods across three language models when the sample budget equals to 16. The evaluation is conducted on three datasets: Obj_C. (Object_Counting), MathQA, and ARC_C. (ARC_Challenge). "Sample budget" refers to the average number of responses sampled per query. The improvements of confidence-augmented methods over their baselines are shown in parentheses. Results for sample budget set to 4 are shown in Appendix E.

# 4 Experiments

## 4.1 Evaluation on Efficient Test-time Scaling

**Experiment Setting.** To ensure a fair comparison across different test-time scaling methods, we use the same sample budgets for each of them. Sample budget refers to the average number of responses each method samples per query. For dynamic methods such as Early Stopping and ASC w/ Conf., we set internal thresholds so that the actual number of samples collected in practice is close to a target budget. To ensure a fair comparison, all methods use responses sampled from Self-Calibration trained models. The experimental details are shown in Appendix C.

**Results.** Table 1 shows the accuracy comparison of different methods with a sample budget of 16. We observe that SC w/ Conf., Early Stopping, and ASC w/ Conf. consistently outperform their base counterparts. On Llama-3.1-8B-Instruct, SC w/ Conf. surpasses SC on MathQA (81.0 to 83.6), while on DeepSeek-R1-Distill-1.5B, Early Stopping outperforms Best-of-N on ARC_challenge (58.9 to 66.5). These results highlight that integrating calibrated confidence enhances test-time scaling with the same sampling budget. More analysis could be found in Appendix B.

## 4.2 Performance Comparison Under Different Sample Budgets

Increasing the sample budget allows for selecting higher-quality outputs but comes at the cost of greater computational expense. To evaluate this trade-off, we compare different methods across multiple sample budgets and visualize their performance trends. As shown in Figure 3, all methods achieve better accuracy as the number of responses increases. Our confidence-guided approaches consistently outperform their original counterparts in most settings. When the sample budget is small, Best-of-N performs better than early stopping because early stopping might stop too soon with a low threshold, missing a better response.

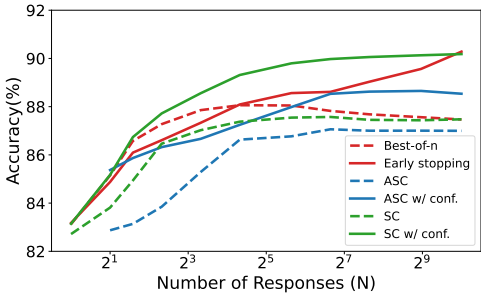



Figure 3: Accuracy over varying sample budgets of different inference strategies on MathQA using self-calibrated Qwen-2.5-7B-Instruction.

# 5 Conclusion

We improve the efficiency of test-time scaling methods in LLMs with reliable confidence estimation. Our Self-Calibration enhances LLM confidence estimation in one forward pass, without requiring any labeled data. We then propose efficient test-time scaling by dynamically adjusting sampling strategies based on calibrated confidence scores. Experiments show that our approaches consistently outperform baselines under the same sample budget. Our findings suggest that reliable confidence estimation and dynamic sampling can substantially enhance the effectiveness and efficiency of repeated sampling.

# References


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

[2] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed H. Chi, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *ArXiv preprint*, abs/2203.11171, 2022.

[3] Afra Amini, Tim Vieira, and Ryan Cotterell. Variational best-of-n alignment. *ArXiv preprint*, abs/2407.06057, 2024.

[4] Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. Do not think that much for 2+3=? on the overthinking of o1-like llms. *ArXiv preprint*, abs/2412.21187, 2024.

[5] Pranjal Aggarwal, Aman Madaan, Yiming Yang, and Mausam. Let's sample step by step: Adaptive-consistency for efficient reasoning and coding with llms. In *Conference on Empirical Methods in Natural Language Processing*, 2023.

[6] Yiwei Li, Peiwen Yuan, Shaoxiong Feng, Boyuan Pan, Xinglin Wang, Bin Sun, Heda Wang, and Kan Li. Escape sky-high cost: Early-stopping self-consistency for multi-step reasoning. *ArXiv preprint*, abs/2401.10480, 2024.

[7] Guangya Wan, Yuqi Wu, Jie Chen, and Sheng Li. Reasoning aware self-consistency: Leveraging reasoning paths for efficient llm sampling. 2024.

[8] Stephanie Lin, Jacob Hilton, and Owain Evans. Teaching models to express their uncertainty in words. *ArXiv preprint*, abs/2205.14334, 2022.

[9] Miao Xiong, Zhiyuan Hu, Xinyang Lu, Yifei Li, Jie Fu, Junxian He, and Bryan Hooi. Can llms express their uncertainty? an empirical evaluation of confidence elicitation in llms. *ArXiv preprint*, abs/2306.13063, 2023.

[10] Jixuan Leng, Chengsong Huang, Banghua Zhu, and Jiaxin Huang. Taming overconfidence in llms: Reward calibration in rlhf. *ArXiv preprint*, abs/2410.09724, 2024.

[11] Ante Wang, Linfeng Song, Ye Tian, Baolin Peng, Lifeng Jin, Haitao Mi, Jinsong Su, and Dong Yu. Self-consistency boosts calibration for math reasoning. In *Conference on Empirical Methods in Natural Language Processing*, 2024.

[12] Katherine Tian, Eric Mitchell, Allan Zhou, Archit Sharma, Rafael Rafailov, Huaxiu Yao, Chelsea Finn, and Christopher D. Manning. Just ask for calibration: Strategies for eliciting calibrated confidence scores from language models fine-tuned with human feedback. *ArXiv preprint*, abs/2305.14975, 2023.

[13] Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zachary 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, Deep Ganguli, Danny Hernandez, Josh Jacobson, John Kernion, Shauna Kravec, Liane Lovitt, Kamal Ndousse, Catherine Olsson, Sam Ringer, Dario Amodei, Tom B. Brown, Jack Clark, Nicholas Joseph, Benjamin Mann, Sam McCandlish, Christopher Olah, and Jared Kaplan. Language models (mostly) know what they know. *ArXiv preprint*, abs/2207.05221, 2022.

[14] Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Anselm Levskaya, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. Efficiently scaling transformer inference. *ArXiv preprint*, abs/2211.05102, 2022.

[15] Katherine Tian, Eric Mitchell, Allan Zhou, Archit Sharma, Rafael Rafailov, Huaxiu Yao, Chelsea Finn, and Christopher D Manning. Just ask for calibration: Strategies for eliciting calibrated confidence scores from language models fine-tuned with human feedback. *ArXiv preprint*, abs/2305.14975, 2023.

[16] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In Doina Precup and Yee Whye Teh, editors, *Proc. of ICML*, volume 70 of *Proceedings of Machine Learning Research*, pages 1321–1330. PMLR, 2017.

[17] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In Doina Precup and Yee Whye Teh, editors, *Proc. of ICML*, volume 70 of *Proceedings of Machine Learning Research*, pages 1321–1330. PMLR, 2017.

[18] Jiaxin Huang, Shixiang Shane Gu, Le Hou, Yuexin Wu, Xuezhi Wang, Hongkun Yu, and Jiawei Han. Large language models can self-improve. *ArXiv preprint*, abs/2210.11610, 2022.

[19] Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V. Le, Christopher R'e, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling. *ArXiv preprint*, abs/2407.21787, 2024.

[20] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed H. Chi, F. Xia, Quoc Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. *ArXiv preprint*, abs/2201.11903, 2022.

[21] Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. In *Proc. of ICLR*. OpenReview.net, 2017.

[22] Paul F. Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 4299–4307, 2017.

[23] Hanze Dong, Wei Xiong, Bo Pang, Haoxiang Wang, Han Zhao, Yingbo Zhou, Nan Jiang, Doyen Sahoo, Caiming Xiong, and Tong Zhang. Rlhf workflow: From reward modeling to online rlhf. *ArXiv preprint*, abs/2405.07863, 2024.

[24] Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. The lessons of developing process reward models in mathematical reasoning. *ArXiv preprint*, abs/2501.07301, 2025.

[25] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, and etc. The llama 3 herd of models. *ArXiv preprint*, abs/2407.21783, 2024.

[26] Qwen Team. Qwen2.5: A party of foundation models, 2024.

[27] DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025.

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

[29] Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *Proc. of NAACL-HLT*, pages 4149–4158, Minneapolis, Minnesota, 2019. Association for Computational Linguistics.

[30] Jian Liu, Leyang Cui, Hanmeng Liu, Dandan Huang, Yile Wang, and Yue Zhang. Logiqa: A challenge dataset for machine reading comprehension with logical reasoning. In Christian Bessiere, editor, *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, pages 3622–3628. ijcai.org, 2020.

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

[32] Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proc. of EMNLP*, pages 2381–2391, Brussels, Belgium, 2018. Association for Computational Linguistics.

[33] Weihao Yu, Zihang Jiang, Yanfei Dong, and Jiashi Feng. Reclor: A reading comprehension dataset requiring logical reasoning. In *Proc. of ICLR*. OpenReview.net, 2020.

[34] Johannes Welbl, Nelson F. Liu, and Matt Gardner. Crowdsourcing multiple choice science questions. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pages 94–106, Copenhagen, Denmark, 2017. Association for Computational Linguistics.

[35] Arkil Patel, Satwik Bhattamishra, and Navin Goyal. Are NLP models really able to solve simple math word problems? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2080–2094, Online, 2021. Association for Computational Linguistics.

[36] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande. *Communications of the ACM*, 64:99 – 106, 2019.

[37] Mirac Suzgun, Nathan Scales, Nathanael Scharli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V. Le, Ed H. Chi, Denny Zhou, and Jason Wei. Challenging big-bench tasks and whether chain-of-thought can solve them. In *Annual Meeting of the Association for Computational Linguistics*, 2022.

[38] Aida Amini, Saadia Gabriel, Shanchuan Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. MathQA: Towards interpretable math word problem solving with operation-based formalisms. In *Proc. of NAACL-HLT*, pages 2357–2367, Minneapolis, Minnesota, 2019. Association for Computational Linguistics.

# A Repeated Sampling

Repeated sampling [19] is a framework that generates multiple responses with Chain-of-Thought prompting [20], then uses a verifier to get the final results. We will introduce three fundamental repeated sampling strategies, which aim to enhance response quality.

## A.1 Best-of-N

For each input query $x$, multiple candidate responses $\{y_i\}$ are sampled, where $1 \leq i \leq N$. A scoring function—such as an additional reward model or a confidence generator—assigns each response a score $c_i = \text{Score}(y_i)$. The simplest selection strategy, known as Best-of-N [1], chooses the response with the highest score as the final answer.

## A.2 Self-Consistency

Self-Consistency [2] (SC) selects the most frequent response among different candidates. Given candidate responses $\{y_1, y_2, \ldots, y_N\}$, the final answer is determined by majority voting. This approach enhances robustness by aggregating diverse model outputs rather than relying on a single highest-scoring response.

## A.3 Adaptive Self-Consistency

Adaptive Self-Consistency (ASC) [5] enhances the standard Self-Consistency approach by dynamically adjusting the number of samples based on agreement among generated responses. This method

| Dataset | Metric | Llama-3.1-8B-Instruct | | Qwen2.5-7B-Instruct | | DS-R1-Distill-1.5B | |
|---|---|---|---|---|---|---|---|
| | | Vanilla | Self-Cal. | Vanilla | Self-Cal. | Vanilla | Self-Cal. |
| *In-Domain Datasets* | | | | | | | |
| GSM8K | ECE ↓ | 13.70 | **3.79** | 87.39 | **16.88** | 46.66 | **40.03** |
| | AUC ↑ | 72.43 | **75.36** | 68.61 | **82.21** | **64.31** | 55.57 |
| | ACC ↑ | 77.44 | **80.43** | **89.41** | 88.74 | 73.38 | **75.36** |
| SVAMP | ECE ↓ | 28.03 | **10.17** | 89.60 | **24.49** | 30.40 | **12.00** |
| | AUC ↑ | 74.17 | **75.79** | 75.10 | **87.46** | 49.33 | **71.27** |
| | ACC ↑ | 72.60 | **75.29** | 90.48 | **92.00** | 52.27 | **57.48** |
| ARC_easy | ECE ↓ | 5.45 | **5.00** | 57.58 | **5.62** | 20.19 | **11.36** |
| | AUC ↑ | **81.16** | 76.89 | 66.10 | **76.75** | 62.89 | **66.86** |
| | ACC ↑ | 87.73 | **89.21** | **92.11** | 92.01 | 54.00 | **56.74** |
| *Out-of-Domain Datasets* | | | | | | | |
| ARC_challenge | ECE ↓ | 7.01 | **6.03** | 55.19 | **10.11** | 11.42 | **5.46** |
| | AUC ↑ | **80.67** | 80.41 | 64.21 | **78.33** | 64.07 | **65.27** |
| | ACC ↑ | 80.87 | **82.38** | **89.37** | 89.05 | 43.39 | **45.77** |
| Object Counting | ECE ↓ | 27.85 | **9.69** | 72.41 | **5.82** | 47.26 | **4.60** |
| | AUC ↑ | 53.84 | **59.47** | 68.07 | **81.02** | 50.39 | **67.61** |
| | ACC ↑ | 60.68 | **65.88** | 72.41 | **74.22** | 55.33 | **58.13** |
| MathQA | ECE ↓ | 12.55 | **8.64** | 62.35 | **18.92** | 13.16 | **4.34** |
| | AUC ↑ | 85.23 | **87.21** | **72.48** | 69.80 | **78.89** | 66.09 |
| | ACC ↑ | 44.18 | **52.34** | 49.85 | **64.18** | 37.69 | **43.21** |

Table 2: Self-Calibration results across both in-domain and out-of domain datasets on three different models. We use Self-Cal. to denote Self-Calibration in the table.

iteratively samples responses and calculates the cumulative frequency $v_k(z)$ and relative frequency $\hat{r}_k(z)$ of each unique answer $z$ after $k$ samples:

$$v_k(z) = \sum_{i=1}^{k} \mathbb{1}(y_i = z), \quad \hat{r}_k(z) = \frac{v_k(z)}{k}. \tag{1}$$

The sampling process continues until the maximum relative frequency $\hat{r}_k(z)$ exceeds a predefined threshold $\tau$. Formally:

$$\begin{cases} k \leftarrow k+1, & \text{if } \max_z \hat{r}_k(z) < \tau, \\ y = \arg\max_z \hat{r}_k(z), & \text{otherwise.} \end{cases}$$

This adaptive strategy reduces computational costs by limiting the number of required samples while maintaining high accuracy in the final answer selection.

# B Analysis

## B.1 Evaluation on Self-Calibration

**Evaluation Metrics.** We evaluate how well our Self-Calibration approach enables models to output accurate confidence estimations. We adopt three standard metrics for evaluating model calibration: ECE, Area Under the Receiver Operating Characteristic Curve (AUC) [21], and accuracy (ACC). The explanation and details of these metrics can be found in Appendix I.

**Results.** In Table 2, we compare our models trained on Self-Calibration objective with their vanilla base models on multiple in-domain and out-of-domain datasets. Self-Calibration trained models consistently lower the ECE score while generally improving accuracy. On GSM8K, Self-Calibration reduces ECE from 13.70 to 3.79 while improving accuracy from 77.44% to 80.43%. Even in cases where ECE slightly increases, such as ARC_easy for Llama-3.1-8B-Instruct, accuracy still improves from 87.73% to 89.21%. Moreover, the strong results on out-of-domain tasks demonstrate the generalizability of our method, as seen in MathQA, where accuracy improves from 49.85% to 64.18% for Qwen2.5-7B-Instruct.

## B.2 Confidence Score Compared to Reward Score from Reward Models

We compare our self-generated confidence scores with established open-source reward model approaches. A reward model is an additional scoring model used to evaluate the quality of generated responses [22]. Deployment of a reward model can introduce several limitations: (1) Reward scores are often unbounded or require dataset-specific normalization, thus difficult to apply a universal threshold for filtering or reweighting responses; (2) Running an extra reward model increases inference time; and (3) A dedicated reward model requires additional GPU memory, and is less efficient for large-scale deployment.

| Model | Dataset | Reward | Confidence |
|---|---|---|---|
| Llama | MathQA | 82.1 | **84.0** |
| | Object Counting | **72.6** | 72.0 |
| | ARC_Challenge | 86.2 | **86.6** |
| Qwen | MathQA | **87.5** | 86.8 |
| | Object Counting | **76.6** | 76.4 |
| | ARC_Challenge | 89.6 | **89.8** |

Table 3: Accuracy of Best-of-16 on two models (Llama-3.1-8B-Instruct and Qwen-2.5-7B-Instruct) on three datasets between self-generated confidence scores and reward scores from additional reward models.

For our analysis, we use the following reward models for comparison: for Llama-3.1-Instruct, we use the reward model from RLHFlow [1] [23]; for Qwen-2.5, we utilize its official Process Reward Model (PRM) [2] [24]. For PRM, we use the lowest reward score across all steps. We ensure the size of each reward model matches with their corresponding base models.

Table 3 shows that our self-generated confidence scores achieve similar performance to reward model scores across all datasets when using Best-of-N. This means that our method, by generating approximately 10 additional tokens, achieves a performance comparable to that of an extra reward model of the same size.

[1] https://huggingface.co/RLHFlow/Llama3.1-8B-ORM-Mistral-Data

[2] https://huggingface.co/Qwen/Qwen2.5-Math-PRM-7B

### B.3 Can Other Confidence Querying Prompts Work Well?

Since our confidence-based approach was trained using a specific confidence querying prompt, we explore whether alternative prompts can achieve similar performance during inference. This analysis is crucial for understanding the robustness of confidence querying prompts different from the training prompt.

We evaluate 6 alternative prompts (listed in Appendix F.1) at inference time. Table 4 shows that despite training with a specific prompt, other prompts yield comparable performance, with only minor variations. This suggests that our confidence querying approach is robust to prompt changes and our training framework improves model calibration rather than overfitting to a specific prompt.

| Dataset | Method | Original | New |
|---|---|---|---|
| MathQA | Early Stopping | 81.7 | $81.52_{\pm 0.30}$ |
| | ASC w/o conf. | 81.9 | $81.80_{\pm 0.21}$ |
| | SC w/o conf. | 82.1 | $81.63_{\pm 0.20}$ |
| Obj_C. | Early Stopping | 67.2 | $70.80_{\pm 1.99}$ |
| | ASC w/o conf. | 74.8 | $74.07_{\pm 1.03}$ |
| | SC w/o conf. | 74.4 | $73.40_{\pm 0.75}$ |
| ARC_C. | Early Stopping | 86.2 | $86.62_{\pm 0.20}$ |
| | ASC w/o conf. | 86.6 | $86.63_{\pm 0.05}$ |
| | SC w/o conf. | 86.4 | $86.35_{\pm 0.25}$ |

Table 4: Accuracy comparison between the original prompt and 6 alternative querying prompts on self-calibrated Llama-3.1B-Instruct. Results are reported as mean$_{\pm \text{std}}$.

### B.4 Ablation Study

We conduct an ablation study to investigate the impact of key components in Self-Calibration, including Dynamic Temperature (EDT), Soft Self-Consistency (SSC), and L1-smooth loss. Table 5 presents our ablation results on the MathQA and Object Counting datasets.

Removing the dynamic temperature or the soft self-consistency score leads to noticeable increases in ECE and/or drops in accuracy. Meanwhile, replacing the L1-smooth objective with MSE achieves slightly lower ECE on MathQA but reduces accuracy on both tasks, suggesting that our chosen loss formulation is more robust overall. These results demonstrate that each module contributes to model calibration and reasoning performance.

| | MathQA | | Object Counting | |
|---|---|---|---|---|
| **Method** | **ECE ↓** | **ACC ↑** | **ECE ↓** | **ACC ↑** |
| ours (full) | 8.64 | 52.34 | 9.69 | 65.88 |
| w/o EDT | 9.14 | 51.44 | 10.40 | 62.88 |
| w/o SSC | 10.85 | 52.18 | 16.02 | 61.12 |
| w/o L1-smooth | 6.43 | 50.86 | 10.48 | 56.48 |

Table 5: Ablation study results on MathQA and Object Counting in Llama-3.1-8B-Instruct. "w/o L1-smooth" means using MSE loss instead of L1-smooth.

## C Experimental Details

**Models.** To evaluate our self-calibration framework and our efficient test-time scaling methods, we conduct experiments on three open-source LLMs: Llama-8B-3.1-Instruct [25], Qwen2.5-7B-Instruct [26] and DeepSeek-R1-Distill-Qwen-1.5B [27]. These models represent diverse architectures and training strategies, allowing us to test the adaptability of our methods.

**Seed Datasets.** We construct our training dataset with diverse reasoning datasets, including: ARC_easy [28], commonsense QA [29], LogiQA [30], GSM8K [31], OpenBookQA [32], ReClor [33], SciQ [34], SVAMP [35] and WinoGrande [36]. For each dataset, we randomly sample 2,000 questions from the training set to construct our training data. Additional details are shown in Appendix H.

**Evaluation Datasets and Prompts.** We evaluate our methods on three benchmark datasets: ARC-Challenge [28], Object-Counting [37] and MathQA [38], covering mathematical and commonsense reasoning tasks in both multiple-choice and open-ended formats. These three datasets are considered out-of-domain as they differ from the datasets used in training, which we refer as in-domain datasets. To evaluate performance in an in-domain setting, we also use the test sets of GSM8K, SVAMP, and ARC_easy. The details of datasets, system prompts and the task prompts of each dataset are shown in Appendix D.

**Baseline Methods.** In addition to the repeated sampling methods mentioned in Sec. A (Best-of-N, SC and ASC), we also include other adaptive test-time scaling methods such as Early-Stopping Self-Consistency (ESC) [6] and Reasoning-Aware Self-Consistency (RASC) [7] for comparison. ESC divides the sampling process into sequential windows and halts further sampling when a high-confidence consensus is reached within a window. RASC enhances sampling efficiency by dynamically evaluating both the generated answers and their corresponding reasoning paths.

## D Prompts

### D.1 Details about Test Dataset

ARC-Challenge includes difficult science questions requiring external knowledge and reasoning. Object-Counting focuses on numerical and spatial reasoning by counting objects. MathQA tests mathematical problem-solving across arithmetic, algebra, and calculus.

### D.2 System Prompt

Here we show the system prompt to let the model generate responses for Chain-of-Thoughts and format for extracting the final results.

```
For the following question, provide a step-by-step explanation
of your thought process.
Use the format demonstrated below for your response.

‘‘‘Example Format:
Explanation: <Your detailed explanation here, outlining how
you arrived at your answer.>
Answer: <Insert your concise answer here, which should
include a {answer_type} (e.g., {demo})>

Ensure that your response strictly adheres to this format.
Explicitly include the words 'Explanation:', 'Answer:'.
```

The answer type includes “option letter” and “number”.

### D.3 Dataset Prompts

We show the prompts for each dataset in Table 6. All datasets and models are open-sourced.

| Dataset | Query Template |
|---|---|
| **gsm8k** | `Question: {question}\n` |
| **sciq** | `Question: {question}\nOptions:\n{options_text}\n` |
| **commonsense_qa** | `Question: {question}\nOptions:\n{options_text}\n` |
| **winogrande** | `Question: {sentence}\nOptions:\nA. {option1}\nB. {option2}\n` |
| **openbookqa** | `Question: {question}\nOptions:\n{options_text}\n` |
| **reclor** | `Passage:\n{passage}\n\nQuestion: {question}\n\nOptions:\n{options_text}\n` |
| **math_qa** | `Problem: {problem_text}\nOptions:\n{options_block}\n` |
| **arc_challenge** | `Question: {question}\nOptions:\n{options_str}\n` |
| **arc_easy** | `Question: {question}\nOptions:\n{options_str}\n` |
| **logiqa** | `Article:\n{context}\n\nQuestion: {question}\n\nOptions:\n{options_text}\n` |
| **svamp** | `Question: {Body + Question}\n` |
| **gpqa** | `{Question}\nOptions:\n{options_text}\n` |
| **aqua_rat** | `Question: {question}\nOptions:\n{options_text}\n` |

Table 6: Query templates for each dataset .

## E Full Main Results

Here we show the main results when sample budget = 4 in Table 7.

When the sample budget is small, the model has limited opportunities to explore different reasoning paths. In this scenario, output variability is often high, and having an additional confidence signal (as in ASC w/ Conf.) is essential for filtering out noisy or incorrect responses. This confidence-augmented method helps select the most promising candidate under tight sampling constraints.

However, when the sample budget increases, the model can generate more candidate solutions, which typically raises the chance of hitting the correct answer. In this setting, Early Stopping

| | Llama-3.1-8B-Instruct | | | Qwen2.5-7B-Instruct | | | DeepSeek-R1-Distill-1.5B | | |
|---|---|---|---|---|---|---|---|---|---|
| Methods | Obj_C. | MathQA | ARC_C. | Obj_C. | MathQA | ARC_C. | Obj_C. | MathQA | ARC_C. |
| Pass@1 | 67.6 | 71.5 | 82.8 | 76.8 | 82.9 | 88.5 | 61.2 | 89.9 | 58.2 |
| *Sample budget = 4* | | | | | | | | | |
| SC | 72.0 | 78.8 | 85.7 | 78.8 | 85.7 | 90.0 | 63.6 | 91.4 | 63.0 |
| SC w/ Conf. | 72.8 (+0.8) | **82.1** (+3.3) | 86.4 (+0.7) | 78.4 (-0.4) | 86.9 (+1.2) | 90.3 (+0.3) | **64.0** (+0.4) | 91.2 (-0.2) | 63.2 (+0.2) |
| Best-of-N | 67.6 | 80.8 | 86.4 | 76.4 | 86.4 | 89.8 | 56.0 | 90.0 | 59.0 |
| Early Stopping | 67.2 (-0.4) | 81.7 (+0.9) | 86.2 (-0.2) | 78.4 (+2.0) | 87.1 (+0.7) | 90.1 (+0.3) | 56.0 (0.0) | 90.6 (+0.6) | 59.0 (0.0) |
| ASC | 74.4 | 80.0 | 86.5 | 79.6 | 86.2 | **91.0** | 61.2 | 91.3 | 63.3 |
| ASC w/ Conf. | **74.8** (+0.4) | 81.9 (+1.9) | **86.6** (+0.1) | **80.0** (+0.4) | **87.2** (+1.0) | 90.6 (-0.4) | 62.8 (+1.6) | **91.6** (+0.3) | **64.6** (+1.3) |
| ESC | 72.0 | 78.6 | 85.8 | **80.0** | 86.9 | 89.6 | 58.0 | 91.2 | 63.0 |
| RASC | 72.4 | 79.0 | 85.8 | **80.0** | 86.4 | 89.8 | 62.6 | 91.2 | 63.1 |

Table 7: Accuracy comparison of different test-time scaling methods across three language models. The evaluation is conducted on three datasets: Obj_C. (Object_Counting), MathQA, and ARC_C. (ARC_Challenge). "Sample budget" refers to the average number of responses sampled per query. The improvements of confidence-augmented methods over their baselines are shown in parentheses. All methods use the same responses generated by Self-Calibration trained models.

approach—especially when coupled with a high confidence threshold—can terminate as soon as it encounters a correct reasoning path.

# F Full Results of Different Confidence Querying Prompts

## F.1 Confidence Querying prompts

We show the 6 confidence querying prompts we used in Sec. B.3.

- $I_1$: Is this the correct answer?
- $I_2$: Does this answer seem right?
- $I_3$: Is this the right answer?
- $I_4$: Is the given answer accurate?
- $I_5$: Would you say this answer is correct?
- $I_6$: Is this response correct?

## F.2 Results of Different Querying Prompts

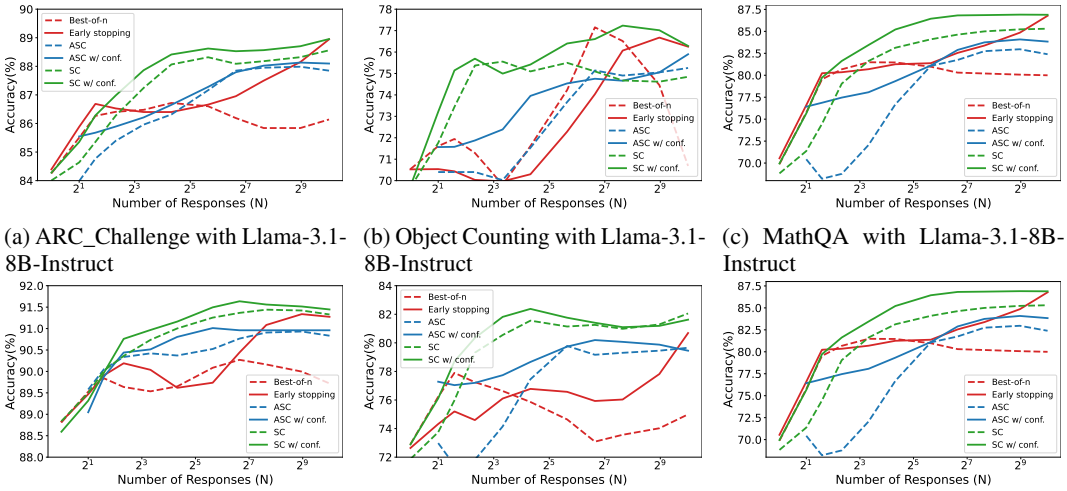
In Table 8, we show the results of different confidence querying prompts for tuned LLama-3.1-8B-Instruct.

| Dataset | Method | 1 | 2 | 3 | 4 | 5 | 6 | Original |
|---|---|---|---|---|---|---|---|---|
| **MathQA** | Early Stopping | 81.7 | 81.4 | 81.7 | 81.3 | 81.1 | **81.9** | 81.7 |
| | ASC w/o conf. | 81.9 | 81.9 | 81.8 | 81.8 | 81.4 | **82.0** | 81.9 |
| | SC w/o conf. | 81.5 | 81.4 | 81.5 | 81.7 | 81.9 | 81.8 | **82.1** |
| **Object_Counting** | Early Stopping | 70.0 | 71.6 | 69.6 | 68.0 | **73.6** | 72.0 | 67.2 |
| | ASC w/o conf. | **73.6** | 73.6 | 74.4 | 73.6 | **76.0** | 73.2 | 74.8 |
| | SC w/o conf. | 72.8 | **74.0** | 73.2 | 72.4 | 74.4 | 73.6 | 72.8 |
| **ARC_challenge** | Early Stopping | **86.8** | 86.4 | **86.8** | 86.5 | **86.8** | 86.4 | 86.2 |
| | ASC w/o conf. | **86.7** | 86.6 | 86.6 | 86.6 | **86.7** | 86.6 | 86.6 |
| | SC w/o conf. | 86.3 | 86.1 | 86.1 | **86.7** | 86.3 | 86.6 | 86.4 |

Table 8: The results for different confidence querying prompt.

# G Results for Different Sample Budgets

Here, we show the performance under different sample budgets of other datasets and models.




(a) ARC_Challenge with Llama-3.1-8B-Instruct (b) Object Counting with Llama-3.1-8B-Instruct (c) MathQA with Llama-3.1-8B-Instruct

(d) ARC_Challenge with Qwen-2.5-7B-Instruction (e) Object Counting with Qwen-2.5-7B-Instruction (f) MathQA with Qwen-2.5-7B-Instruction

Figure 4: Performance comparison of different inference strategies across datasets and models using Self-Calibration.

# H Hyperparameters

This section details the hyperparameters used in our experiments. We categorize them into training data generation, training process, and response generation

## H.1 Training Data Generation

When creating the datasets, we set the number of responses for each query $N = 32$. For the parameter in dynamic temperature, we follow the default hyperparameter settings from the original paper: $T_0 = 0.8$, $M = 0.8$, $\gamma = 1.0$, and $\tau_0 = 0.001$.

## H.2 Training Process

In the training objective, we set the threshold $\eta = 0.75$ to filter the response used in generation ability training and the weight $w = 0.1$ to balance two losses.

In the training process, we use the AdamW optimizer with a learning rate of $5 \times 10^{-5}$. The total number of training samples is set to 100,000, while 1,000 samples are used for evaluation. We employ a batch size of 1 with gradient accumulation steps of 64 to simulate a larger effective batch size. The model is trained for 1 epoch.

For parameter-efficient fine-tuning, we apply LoRA with rank $r = 32$, scaling factor $\alpha = 16$, and dropout rate of $0.05$. In the whole training examples, the ratio of causal language modeling data is 0.7. We train the model on multiple datasets with varying proportions of training and evaluation data. Specifically, GSM8K and SVAMP each contribute 15% of the training and evaluation samples. SciQ, CommonsenseQA, Winogrande, OpenBookQA, ReClor, ARC-Easy, and LogiQA each contribute 5% of the training and evaluation samples.

During the sample training data selection process, we ensure that the data is evenly distributed across different confidence intervals. This balancing strategy prevents overrepresentation of any specific confidence range, allowing the model to learn from a diverse set of samples. By maintaining an equal number of training examples in each confidence bin, we improve the robustness of confidence calibration and reduce potential biases in the learning process.

## H.3 Response Generation

When generating the response, we set the temperature equals to 1.0.

# I Explanation of Calibration Estimation

ECE measures the discrepancy between a model's predicted confidence and its actual accuracy, defined as:

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

where $M$ is the number of bins, $B_m$ represents the set of samples in the $m$-th bin, and $N$ is the total number of samples. A lower ECE value indicates better calibration, meaning the model's confidence aligns more closely with its actual correctness. In our experiments we set $m = 10$.