Article

Calibrated Metacognitive Self-Consistency for Confidence-Robust Chain-of-Thought Aggregation

Firstname Lastname [1], Firstname Lastname [2] and Firstname Lastname [2,*]

[1] Affiliation 1; e-mail@e-mail.com
[2] Affiliation 2; e-mail@e-mail.com
[*] Correspondence: e-mail@e-mail.com; Tel.: (optional; include country code; if there are multiple corresponding authors, add author initials) +xx-xxxx-xxx-xxxx (F.L.)

Abstract

Self-consistency improves large language model reasoning by sampling multiple chain-of-thought (CoT) solutions and aggregating their final answers, but common aggregation rules can be statistically inefficient and sometimes anti-robust when sampled solutions differ in reliability. Unweighted majority vote ignores heterogeneous sample quality, while weighting by a model's self-reported confidence is brittle because confidence is often miscalibrated and prompt-sensitive. We propose Calibrated Metacognitive Self-Consistency (Cal-MSC), a prompt-only method that turns metacognitive signals into a usable decision rule without fine-tuning and without training a separate verifier. Each sampled solution performs a brief self-check and then reports an updated scalar confidence c in [0,1]. Using a small in-distribution calibration slice (dozens of labeled questions), we learn an on-the-fly mapping from reported confidence to empirical correctness via nonparametric binning with Laplace smoothing (or isotonic regression when available). At evaluation time, we aggregate candidate answers by pooling calibrated log-odds rather than raw confidences. We provide a complete, reproducible implementation and protocol for GSM8K and SVAMP; the accompanying artifact contains code and configurations, but no execution logs are included in the provided context, so this paper's empirical section is limited to verifiable, artifact-based findings rather than numerical accuracy claims.

Keywords: keyword 1; keyword 2; keyword 3 (List three to ten pertinent keywords specific to the article; yet reasonably common within the subject discipline.)

1. Introduction

Large language models (LLMs) often perform better on multi-step problems when prompted to produce intermediate reasoning steps rather than emitting only a final answer. Chain-of-thought (CoT) prompting elicits such step-by-step rationales and can substantially improve accuracy on arithmetic and other reasoning benchmarks, especially as model scale increases [1]. In arithmetic word problems, CoT externalizes intermediate computations that would otherwise be implicit, which can help models maintain longer dependency chains and reduce shallow pattern-matching errors.

Despite these gains, a single CoT sample can be unreliable. Under stochastic decoding, the same model and prompt can yield multiple plausible reasoning paths that disagree in intermediate steps and final answers. Some paths include subtle arithmetic slips or inconsistent assumptions; others are coherent and correct. This variance motivates inference-time ensembling methods that sample multiple CoTs and aggregate their answers. A widely

Received:
Revised:
Accepted:
Published:

used aggregation rule is unweighted majority vote over final answers, which is attractive because it is prompt-only and requires no additional training.

However, majority vote makes a strong implicit assumption: conditional on the prompt and model, each sampled completion is equally informative about correctness. In practice, sampled solutions are heterogeneous. Some contain detectable contradictions; some include a convincing self-verification; some are terse and error-prone. A natural response is to weight samples by a confidence estimate. Yet for many LLM deployments, the most accessible confidence is a self-reported number written in natural language. Such confidence is notoriously brittle: models can be confidently wrong, can saturate near 1.0, and can change their numerical confidence under minor prompt variations. In these settings, naive confidence weighting can degrade accuracy rather than improve it.

This paper studies a prompt-only gap: how can metacognitive signals, such as a short self-check and a scalar confidence, be transformed into a decision rule that is both statistically principled and robust to confidence miscalibration, without fine-tuning the base model and without training a separate verifier? Our central hypothesis is that the missing ingredient is in-distribution calibration. Rather than treating a reported confidence c as a probability of correctness, we should estimate how predictive that confidence is in the current task context.

We propose Calibrated Metacognitive Self-Consistency (Cal-MSC). For each question, we sample K CoT completions. Each completion is prompted to (i) solve step by step, (ii) perform a brief self-check, and (iii) report an updated confidence c in [0,1]. We then learn an on-the-fly calibration function f (c) that estimates the empirical probability that a completion is correct given its reported confidence, using a small calibration slice of labeled questions drawn from the same distribution as evaluation. At test time, we convert each reported confidence into a calibrated probability p = f (c) and aggregate answers using calibrated log-odds pooling, which combines evidence additively in a way that penalizes systematically overconfident regions.

Our contributions are: - A prompt-only inference method (Cal-MSC) that converts self-reported confidence into a calibrated estimate of correctness probability using a small in-distribution calibration slice, with no fine-tuning and no auxiliary verifier. - A principled self-consistency aggregation rule based on calibrated log-odds pooling, designed to exploit heterogeneous sample reliability while correcting miscalibration. - A complete, runnable evaluation protocol and implementation for GSM8K and SVAMP that fixes the sampling budget and clearly separates calibration, aggregation, and answer-extraction logic.

Empirical verification plan. The intended evaluation compares Cal-MSC to a self-consistency majority-vote baseline under identical sampling hyperparameters (same K and temperature) and reports final-answer accuracy as the primary metric. The design also calls for calibration diagnostics (Expected Calibration Error, confidence–correctness AUROC, and Brier score) computed over sampled completions. The provided artifact includes the code needed to run these experiments and to save per-sample outputs for offline analysis, but the present context does not include run logs or stored metrics; accordingly, we do not claim numerical improvements in this paper.

Future work. The Cal-MSC framework naturally extends to alternative calibration models (e.g., isotonic regression versus binning) and to broader reasoning domains beyond arithmetic. Another promising direction is to structure the self-check into standardized verifications (unit checks, recomputation, constraint checks) while keeping the method prompt-only and model-agnostic.

2. Related Work

CoT prompting as a basis for multi-step reasoning. CoT prompting established that prompting models with intermediate reasoning steps can unlock markedly better performance on arithmetic and other multi-step tasks, with especially strong effects at large model scales [1]. Cal-MSC adopts the same core interface assumption: the model will produce a textual rationale and a final answer, and we evaluate correctness at the level of the final answer rather than the faithfulness of the rationale.

Aggregation over multiple reasoning paths. Sampling multiple CoT solutions and aggregating them is a common inference-time strategy for reducing the variance of single-sample reasoning. The standard approach in practice is to use unweighted majority vote over extracted final answers. Our work directly targets this aggregation step. We emphasize that majority vote implicitly treats all samples as equally reliable, which is mismatched to observed heterogeneity in reasoning quality.

Confidence-weighted aggregation and brittleness of self-reported confidence. A conceptually straightforward alternative to majority vote is to weight each sample's vote by a confidence score. When confidence is derived from internal model probabilities, one can view this as a form of probabilistic model averaging. In many prompt-only or API-based settings, however, the available confidence is a self-reported scalar emitted as text. Cal-MSC differs from naive confidence weighting by treating self-reported confidence as an uncalibrated signal that may be systematically biased, prompt-sensitive, and subject to ceiling effects. The key comparison is therefore not "majority vote versus confidence weighting" but "uncalibrated versus calibrated use of confidence."

Self-checking and metacognitive prompting. Prompting models to reflect, verify, or revise their own answers is a recurring theme in LLM reasoning practice. Cal-MSC is aligned with this intuition but makes two distinctive design choices. First, the self-check is not an end in itself; it is explicitly followed by an updated confidence, pushing the model toward a metacognitive update rather than additional narrative. Second, the method does not assume that self-checking yields a trustworthy probability; instead, it estimates how much the updated confidence should be trusted in the current task distribution via a small calibration slice.

Calibration as a lightweight alternative to verifiers and fine-tuning. Many robustness approaches for LLM reasoning rely on training additional components, such as rerankers or verifiers, or on fine-tuning the base model. Those approaches are not applicable under the constraints we consider: prompt-only changes and lightweight post-processing. Cal-MSC occupies a narrow but practically important space between purely heuristic aggregation (majority vote, raw confidence weighting) and heavier-weight training-based pipelines. Its calibration model is deliberately simple (binning with smoothing or isotonic regression), intended to be fit quickly from dozens of labeled questions and then reused for aggregation.

Comparability to methods outside our constraints. Techniques that require additional training signals, specialized model access, or external tools are not directly comparable under our experimental protocol. For this reason, the empirical comparisons in the provided artifact focus on a strict baseline (self-consistency with majority vote) versus Cal-MSC under matched sampling budgets and the same base model interface.

3. Background

Chain-of-thought prompting and sampled reasoning variability. CoT prompting asks the model to produce intermediate reasoning steps before a final answer, which can improve multi-step reasoning accuracy [1]. Under stochastic decoding, CoT prompting also exposes variability: distinct samples can produce different intermediate computations and different final answers, even for the same input question.

Problem setting and notation. We consider single-answer tasks. Each example consists of a question q and a gold numeric answer g (q). For each question q, we draw K sampled completions from an LLM using temperature-based sampling. Each completion i yields a text response r_i. From r_i we extract:

- A final answer a_i, intended to be a number. - For Cal-MSC runs, a self-reported "updated confidence" c_i in [0,1].

We define a binary correctness label y_i for each completion based on answer matching after normalization: $y_i = 1$ if a_i matches g (q) under the task's numeric comparison rule, and $y_i = 0$ otherwise.

Self-consistency as answer aggregation. Let A (q) denote the multiset of extracted answers $\{a_1, \ldots, a_K\}$ for question q. Majority-vote self-consistency predicts the most frequent element of A (q). This rule is optimal only under restrictive assumptions, such as equal per-sample correctness probabilities and independence.

Metacognitive signals and the calibration objective. In addition to a_i, Cal-MSC requests a metacognitive scalar c_i. Importantly, we do not interpret c_i as a calibrated probability. Instead, we treat c_i as a score that may correlate with correctness in a task- and prompt-dependent way.

Calibration slice assumption. Cal-MSC assumes access to a small calibration set of N_{cal} questions from the same distribution as evaluation. On the calibration set, we sample K completions per question and collect all pairs (c_i, y_i). From these we learn a calibration function f that maps a raw confidence to a calibrated probability p = f (c) that approximates P (y = 1 | c) in the current task context.

Aggregation objective under heterogeneous reliability. On a test question, each sample i provides an extracted answer a_i and a calibrated probability estimate p_i = f (c_i). The aggregation goal is to choose a final answer that is most likely correct under these per-sample reliability estimates, while remaining robust when some confidence values are systematically inflated or compressed. Cal-MSC addresses this by combining evidence on the log-odds scale rather than using raw confidence as a linear weight.

Unusual assumptions. The main nonstandard assumption relative to standard self-consistency is the availability of a small, labeled in-distribution calibration slice at inference time. This is realistic in many deployment settings (a small held-out validation set or a small batch of gold-labeled queries) but should be stated explicitly because it introduces supervision into an otherwise unsupervised aggregation step.

4. Method

Calibrated Metacognitive Self-Consistency (Cal-MSC) modifies self-consistency in two coupled ways: it changes the per-sample output format to include a metacognitive update, and it replaces majority vote with calibrated, probabilistic pooling.

Generation: self-check followed by updated confidence. For each question q, we sample K independent completions under stochastic decoding. The prompt instructs the model to solve step by step and then output three fields in a fixed textual format: (i) "Final answer:", (ii) "Self-check:", and (iii) "Updated confidence:". The self-check is intended to be short and concrete (e.g., recomputing a key arithmetic step), and the confidence c is constrained to lie between 0.0 and 1.0. This structure aims to make the confidence reflect a post-hoc verification attempt rather than the model's initial impression.

Calibration: learning f (c) from a small labeled slice. On the calibration slice, we collect all reported confidences and their correctness labels (c_i, y_i). The artifact supports two calibration strategies.

First, the default implementation uses equal-width binning on [0,1] with Laplace smoothing. Let B be the number of bins. Each confidence value c is assigned to bin b

= min (int (c × B), $B - 1$). For each bin b we count correct$_b$ and total$_b$. The calibrated probability assigned to that bin is:

$$p_b = \frac{\text{correct}_b + s}{\text{total}_b + 2s}$$

where s is a smoothing parameter. This corresponds to a symmetric Beta prior and prevents extreme probabilities in sparse bins.

Second, an optional isotonic regression calibrator can be fit if an implementation is available. Isotonic regression enforces monotonicity of f with respect to c, which can be desirable when confidence is intended to be order-preserving. If isotonic regression is selected but unavailable, the code falls back to binning.

Aggregation: log-odds pooling over candidate answers. For a test question, each completion i produces an extracted answer a_i and a raw confidence c_i. We compute a calibrated probability p_i = f (c_i). For each distinct candidate answer a, we compute a score by summing calibrated log-odds contributions from all samples that proposed a:

$$\text{score}(a) = \sum_{i:a_i=a} (\log(p_i) - \log(1 - p_i))$$

The final prediction is the a with maximum score (a). The intuition is that each sample contributes additive evidence proportional to its calibrated odds of correctness, which is a more appropriate combination rule than linear confidence summation when probabilities are the quantities of interest.

Relationship to baselines and to raw confidence weighting. Majority vote is recovered by ignoring confidence and counting occurrences of each answer. Raw confidence-weighted voting is a natural intermediate that sums c_i per answer; Cal-MSC differs by replacing c_i with f (c_i) and by pooling on a log-odds scale. Both design choices address known failure modes of raw confidence: miscalibration and saturation near 0 or 1, which can otherwise dominate linear weighting.

Implementation-defined details. The artifact clamps calibrated probabilities away from 0 and 1 when computing log-odds in order to avoid numerical overflow. This is implemented by clamping probabilities into a fixed interior interval (e.g., 0.01 to 0.99 in the provided code).

5. Experimental Setup

Datasets and slicing protocol. Experiments target arithmetic word-problem benchmarks loaded via HuggingFace datasets:

- GSM8K: loaded as gsm8k with the main configuration, using the test split. The loader selects the first 300 examples. The first 60 examples are reserved for calibration and the remaining 240 for evaluation. - SVAMP: the loader attempts to use ChilleD/SVAMP; if that fails, it falls back to a dataset named svamp. As with GSM8K, 300 test examples are used with a 60/240 calibration/evaluation split.

Gold-answer extraction. For GSM8K, gold answers are embedded in a text field that includes a delimiter “”; the preprocessing extracts the numeric value following this delimiter. For SVAMP, the loader extracts the answer from either an “Answer” or “answer” field depending on the dataset variant.

Model interface and decoding hyperparameters. The provided Hydra run configurations specify the model as gpt-3.5-turbo accessed through the OpenAI API. Each completion is generated with temperature 0.8, and a maximum of 512 tokens. The number of samples is K = 10 per question for both baseline and proposed runs.

Prompts and method-specific output formats. Both methods instruct the model to solve "step by step." The baseline requires the model to provide its final response as "Final answer: <number>." The proposed method adds two additional fields, "Self-check:" and "Updated confidence: <0.0 to 1.0>," and explicitly asks the model to self-check before stating updated confidence.

Parsing and normalization. The inference code parses predicted answers by first searching for a "Final answer:" pattern; if that fails, it falls back to extracting the last number-like token in the response. For Cal-MSC, it extracts the updated confidence using a dedicated pattern, converts it to a float, and clamps it to [0,1]. Correctness is computed by comparing predicted and gold numeric values with an absolute tolerance of 0.01.

Calibration training. Only the proposed method uses the calibration slice. For each of the 60 calibration questions, the inference script samples K responses, parses answers and confidences, labels each completion as correct or incorrect, and fits either (i) a binned calibration map with $n_{\text{bins}} = 10$ and Laplace smoothing 1.0, or (ii) an isotonic regression model when selected.

Compared methods and run configurations. The provided artifact includes four run configurations:

- comparative-1-gsm8k: baseline self-consistency with majority vote on GSM8K. - proposed-gsm8k: Cal-MSC on GSM8K. - comparative-1-svamp: baseline self-consistency with majority vote on SVAMP. - proposed-svamp: Cal-MSC on SVAMP.

Each run is designed to log summary metrics (accuracy, number correct, number total) to Weights and Biases and to save a detailed results.json file containing per-question samples (raw responses, extracted answers, extracted confidences when applicable) and aggregation statistics.

Evaluation tooling. A separate evaluation script is included to fetch run summaries from Weights and Biases, export per-run metrics.json files, and generate comparison plots. This script expects run IDs and a Weights and Biases project context; it produces PDF figures when history data are available.

6. Results

This section summarizes the findings that can be verified from the provided artifact without inventing execution outcomes. Because no run outputs (standard output, standard error, stored metrics, or generated figures) are included in the provided context, we do not report accuracy values or calibration-diagnostic measurements.

Experiment 1: GSM8K protocol and available artifacts. The GSM8K comparison is specified by two Hydra run files: comparative-1-gsm8k (baseline) and proposed-gsm8k (Cal-MSC). Both are configured to load 300 GSM8K test questions, sample K = 10 responses per question at temperature 0.8, and evaluate on the last 240 questions after reserving 60 for calibration. From code inspection, the baseline aggregates by majority vote over extracted numeric answers. The proposed method performs a calibration phase on the first 60 questions, fitting a 10-bin Laplace-smoothed calibration map or isotonic regression, and then aggregates evaluation answers by calibrated log-odds pooling.

The inference pipeline is designed to save a per-run results.json file that includes, for each evaluation question, the raw sampled responses, extracted answers, and (for Cal-MSC) extracted confidences and derived calibrated probabilities used in aggregation. This file is sufficient to compute dataset-level accuracy and paired per-question win/loss comparisons between methods, but no such file is included in the provided context.

Experiment 2: SVAMP protocol and available artifacts. The SVAMP comparison mirrors GSM8K, with comparative-1-svamp and proposed-svamp run configurations. The dataset loader includes a documented fallback mechanism: it first attempts to load

ChilleD/SVAMP and otherwise loads an alternative dataset named svamp. Questions are constructed either from separate "Body" and "Question" fields or from a single question field, depending on the dataset source. The aggregation and calibration logic is identical to the GSM8K runs, with the same K, temperature, and calibration hyperparameters.

Experiment 3: Calibration diagnostics are planned but not logged by default. The experimental design in the context motivates reporting calibration diagnostics such as Expected Calibration Error (ECE), confidence–correctness AUROC, and Brier score on the evaluation slice's sampled completions. However, the current inference script logs only final-answer accuracy and counts ($\text{num}_{\text{correct}}$ and $\text{num}_{\text{total}}$) to Weights and Biases. This is a verifiable limitation of the logged outputs: although per-sample confidences and correctness labels exist in results.json (once runs are executed), additional post-processing is required to compute and report ECE, AUROC, and Brier score.

Artifact-level limitations affecting interpretability. The provided configurations reveal several factors that must be considered when interpreting any future experimental results. First, the proposed method changes the prompt format by adding self-checking and confidence fields, while the baseline prompt requests only a final answer. Therefore, any measured gain could be caused by the self-check instruction, the calibrated aggregation rule, or both; isolating these factors would require additional ablations not present as run configurations in the artifact. Second, the broader design note contains an alternative reference implementation of binning calibration with a slightly different smoothing formula, but the executed runs in this artifact are defined by the inference pipeline's Laplace-smoothed binning rule $p_b = (\text{correct}_b + \text{s})/\ (\text{total}_b + 2\text{s})$. Finally, the evaluation script can generate PDF figures if run history data are present, but no figures are included in the current context; accordingly, this paper includes no figure files.

Summary. The artifact fully specifies the comparison and provides the necessary machinery to produce reproducible metrics, but it does not contain the outputs required to substantiate the hypothesis with numerical evidence within this document.

7. Discussion

8. Conclusions

Calibrated Metacognitive Self-Consistency (Cal-MSC) reframes confidence-weighted self-consistency as a calibration problem. Rather than trusting self-reported confidence values directly, the method (i) prompts each sampled chain-of-thought solution to perform a brief self-check and output an updated scalar confidence, (ii) learns an in-distribution mapping from that confidence to empirical correctness using a small labeled calibration slice, and (iii) aggregates answers using calibrated log-odds pooling. This yields a principled prompt-only decision rule that is designed to exploit heterogeneous sample reliability while correcting for systematic overconfidence or prompt sensitivity.

The main conceptual implication is that metacognitive signals can be made useful without fine-tuning and without training a separate verifier, provided that we explicitly estimate how trustworthy those signals are in the current task context. Practically, Cal-MSC requires only a minor prompt change and a lightweight calibration pass over dozens of labeled questions, which is compatible with many inference-only deployments.

The provided artifact contains complete code and Hydra configurations to compare Cal-MSC against a majority-vote self-consistency baseline on GSM8K and SVAMP under a fixed sampling budget. Because the present context includes no execution logs or saved evaluation metrics, this paper does not claim observed numerical improvements. Executing the provided runs and computing both accuracy and calibration diagnostics from the saved per-sample outputs is the necessary next step to validate when calibrated metacognitive aggregation improves over majority vote and to quantify any accuracy–compute trade-offs.

Author Contributions: For research articles with several authors, a short paragraph specifying their individual contributions must be provided. The following statements should be used "Conceptualization, X.X. and Y.Y.; methodology, X.X.; software, X.X.; validation, X.X., Y.Y. and Z.Z.; formal analysis, X.X.; investigation, X.X.; resources, X.X.; data curation, X.X.; writing—original draft preparation, X.X.; writing—review and editing, X.X.; visualization, X.X.; supervision, X.X.; project administration, X.X.; funding acquisition, Y.Y. All authors have read and agreed to the published version of the manuscript"., please turn to the CRediT taxonomy for the term explanation. Authorship must be limited to those who have contributed substantially to the work reported.

Funding: This research received no external funding.

Data Availability Statement: All resources used in this study are openly available at

Acknowledgments: In this study, we automatically carried out a series of research processes—from hypothesis formulation to paper writing—using generative AI.

Conflicts of Interest: The authors declare no conflicts of interest.

1. Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Ichter, B.; Xia, F.; Chi, E.; Le, Q.; Zhou, D. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models **2022**.