Article

Falsify, Backsolve, and Vote: Prompt-Only Self-Consistency Under Correlated Reasoning Errors

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 (SC) improves chain-of-thought reasoning by sampling multiple solutions and majority-voting the final answer, implicitly treating samples as independent evidence for correctness [1]. In arithmetic word problems, however, many samples can share the same hidden mistake (for example, a repeated misread of quantities or a systematic arithmetic slip), producing a high-agreement but wrong majority. We study whether this failure mode can be mitigated using prompt-time interventions only, without training an external verifier. We propose Falsification-Augmented Bidirectional Self-Consistency (FAB-SC), which augments each sampled solution with two structured checks run by the same base model: (1) a backsolve or plug-back verification that tests the proposed answer against constraints extracted from the problem, and (2) a small ensemble of adversarial refuter prompts that attempt to falsify the solution. We combine these signals with an explicit length penalty to reduce verbosity-driven confidence artifacts and use the resulting reliability weights for importance-weighted voting. Using GPT-4o, we evaluate on pilot slices of GSM8K (200 test items) and SVAMP (300 test items). FAB-SC yields mixed accuracy changes (0.785 vs 0.810 on GSM8K; 0.853 vs 0.843 on SVAMP) while increasing runtime by about 18×, clarifying both the potential and the compute cost of prompt-only verification.

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

Chain-of-thought (CoT) prompting demonstrated that asking large language models (LLMs) to produce intermediate natural-language reasoning can substantially improve performance on multi-step tasks, including grade-school arithmetic word problems [2]. CoT is appealing because it is training-free and often turns a single model into a more capable reasoner simply by changing the inference-time interface. However, CoT also highlights an important reliability tension: intermediate rationales are not guaranteed to be correct or faithful, and with greedy decoding a single early mistake can propagate deterministically to an incorrect final answer.

Self-Consistency (SC) is one of the most widely adopted training-free mitigations for this brittleness [1]. Instead of decoding one rationale, SC samples multiple rationales under stochastic decoding and aggregates their final answers, typically using an unweighted majority vote. When sampling produces sufficiently diverse reasoning traces, the correct answer frequently receives more support than any single incorrect alternative, so agreement

Received:
Revised:
Accepted:
Published:

https://doi.org/10.3390/1010000

among samples becomes an effective unsupervised selection signal. In many benchmarks, this simple “self-ensemble” yields large gains over greedy CoT decoding.

This paper examines a specific SC failure mode that becomes salient precisely in settings where users expect SC to act as a robustness mechanism: correlated reasoning errors. The central assumption behind majority voting is not literally independence, but it does rely on the idea that sampled rationales provide meaningfully different evidence. In arithmetic word problems, LLM samples often share a common hidden mistake: misreading a quantity, applying the wrong operation consistently across samples, or silently dropping a constraint. In such cases the induced answer distribution can be sharply peaked at an incorrect value. SC can then converge to a high-agreement wrong majority, and increasing the sample count can strengthen the wrong consensus rather than correct it. This failure mode is practically hazardous: a wrong answer supported by many fluent rationales can be more persuasive than a single wrong answer, especially in education and decision support.

A natural response under prompt-only constraints is to add self-evaluation: ask the model to critique or score its own solutions. However, using a single self-critique score per sampled rationale is vulnerable to two limitations that are closely aligned with correlated-error regimes. First, the same model that produced the error is also the evaluator, so systematic biases can persist during evaluation (“grading its own homework”). Second, self-scores can correlate with surface features such as fluency or length rather than with validity, potentially overweighting verbose but wrong rationales.

We therefore ask whether SC can be made more robust to correlated mistakes using only inference-time prompting, by approximating two common human checking habits: (i) verifying an answer by backsolving or plugging it back into the stated constraints, and (ii) attempting to falsify a proposed solution by actively searching for a flaw. This yields our method, Falsification-Augmented Bidirectional Self-Consistency (FAB-SC). FAB-SC retains SC’s forward sampling stage, but augments each sampled rationale with two structured checks run by the same base model: a backsolve or plug-back constraint verification prompt (B-check) and a small ensemble of adversarial refuter prompts (F-check). We combine these signals with a deterministic length penalty computed outside the model to reduce verbosity-as-confidence effects, and we aggregate answers using importance-weighted voting rather than unweighted majority.

We evaluate FAB-SC in a controlled pilot setting where both FAB-SC and SC use the same base model (GPT-4o) and the same forward-sampling budget (8 forward samples per question). We test on pilot slices of two established arithmetic word-problem benchmarks used in prior CoT and SC studies: GSM8K and SVAMP [1,2]. On the first 200 GSM8K test items, FAB-SC decreases accuracy relative to SC (0.785 vs 0.810) while increasing wall-clock runtime from 1,164 seconds to 20,587 seconds. On the first 300 SVAMP test items, FAB-SC slightly improves accuracy (0.853 vs 0.843) but increases runtime from 1,548 seconds to 27,764 seconds. These outcomes indicate that prompt-only verification signals can sometimes help, but under our tested configuration they are not consistently beneficial and impose a large compute cost.

Contributions: - We identify and formalize a prompt-only robustness objective for self-consistency: mitigating wrong-majority failures caused by correlated reasoning errors without training an external verifier. - We propose FAB-SC, a training-free inference procedure that combines bidirectional constraint verification (backsolve/plug-back) and an adversarial falsification ensemble into an importance-weighted self-consistency aggregator, augmented by an explicit anti-verbosity regularizer. - We provide a concrete, reproducible implementation pattern (Hydra-configured inference, deterministic numeric parsing, and structured verification prompts) and report pilot results on GSM8K and SVAMP with GPT-4o, including both accuracy and wall-clock runtime.

The mixed pilot results highlight several future directions. Because FAB-SC multiplies inference cost, budget-aware policies are likely necessary: verification could be allocated adaptively, and early stopping ideas developed for reducing SC's sampling cost offer a natural foundation for compute-aware verification pipelines [3]. In addition, mechanism-targeted diagnostics (for example, how often FAB-SC flips an SC wrong majority) are needed to test whether the method addresses its intended failure mode beyond aggregate accuracy.

2. Related Work

CoT prompting established that few-shot exemplars with intermediate reasoning steps can unlock multi-step problem solving in LLMs without parameter updates [2]. Our work operates in the same training-free regime but focuses on what to do after CoT produces multiple plausible rationales: how to aggregate them when agreement is potentially misleading.

Self-Consistency (SC) is the closest precursor and the primary baseline [1]. SC replaces greedy decoding with sample-and-marginalize: generate multiple reasoning paths and select the final answer that receives the most support, most commonly via unweighted majority vote. SC also explored probability-based weighting (length-normalized sequence likelihood) as an alternative to voting. FAB-SC is designed as a direct descendant of SC: it uses the same forward sampling mechanism and the same assumption that a single final numeric answer can be parsed, but it changes the evidential model. Rather than treating each sample as one equal vote or relying on the model's generative probability as a confidence proxy, FAB-SC explicitly induces verification behaviors at prompt time and uses their outputs to weight votes.

A key difference is the type of signal used for aggregation. SC's majority vote leverages agreement among sampled answers; likelihood weighting leverages the model's own token-level probabilities [1]. Both can fail when the model is confidently wrong in a correlated way. FAB-SC instead seeks signals that are conditionally orthogonal to the forward solve. Backsolve or plug-back verification changes the direction of reasoning from "derive the answer" to "test the answer against constraints," which is closer to constraint satisfaction checking than to re-generation. Adversarial falsification reframes evaluation as a search for negative evidence, encouraging the model to actively look for misread quantities, arithmetic slips, or missing constraints. While these checks still use the same model and therefore cannot fully eliminate shared biases, they aim to reduce reliance on a single self-confidence channel.

Compute cost is a well-known limitation of SC, since inference-time budget scales with the number of sampled paths [1]. Recent work has studied early stopping for SC to reduce sampling cost when additional samples are unlikely to change the final answer [3]. FAB-SC amplifies the compute concern because it adds multiple verification calls per forward sample. This makes compute-aware evaluation and budget allocation central to the practicality of verification-augmented SC: any accuracy gain must be interpreted relative to the added inference cost.

In summary, FAB-SC extends SC by replacing "agreement implies correctness" with "answers that survive constraint checking and adversarial refutation should receive more weight." It remains within the prompt-only, single-model setting of SC [1], but targets correlated wrong-majority regimes where unweighted voting and simple confidence proxies can be brittle.

3. Background

Arithmetic word problems offer a controlled setting for evaluating multi-step reasoning under prompt-time interventions because each item typically has a single intended numeric answer. GSM8K is a widely used benchmark of grade-school math problems and has become a standard testbed for chain-of-thought prompting [2]. SVAMP is a related dataset designed to probe robustness to changes in problem phrasing and structure while preserving similar underlying arithmetic reasoning; it is often evaluated alongside GSM8K in self-consistency work [1]. In both datasets, success can be measured by parsing a final numeric answer from the model output and comparing it to the provided gold answer.

Problem setting and notation: We consider a dataset of N problem instances. Each instance j consists of a question string q_j and a gold numeric answer y_j. Given an LLM and a fixed CoT prompt template, we generate m forward samples for a given question q_j using stochastic decoding. Each forward generation i produces a text rationale $r_{j,i}$. A rule-based parser extracts a candidate numeric answer $a_{j,i}$ from $r_{j,i}$. In our implementation, extraction searches for patterns such as "The answer is X", "Therefore, X", "#### X", or "Final answer: X", and otherwise falls back to the last number in the output.

Self-Consistency as a baseline aggregator: For SC, the aggregated prediction for question j is the most frequent parsed value among $a_{j,i_{i=1..m}}$ [1]. SC's effectiveness relies on the empirical regularity that correct answers often have higher sample agreement than incorrect answers, which can be dispersed across many wrong values when reasoning errors vary.

Correlated errors and wrong-majority behavior: The agreement heuristic breaks down when multiple sampled rationales share a latent mistake. In arithmetic word problems, correlated errors can arise from consistently misreading a quantity, systematically applying the wrong arithmetic operation, or ignoring a constraint that appears in the natural language. When this happens, the distribution of sampled answers can concentrate on a single incorrect value. SC's majority vote then becomes overconfident in the wrong answer because it has no mechanism to distinguish "agreement due to correctness" from "agreement due to a shared misunderstanding."

Prompt-time verification under a single-model constraint: Under the constraint of no fine-tuning and no external verifier, verification must be performed by the same base model at inference time. This is intrinsically challenging: the evaluator can reproduce the generator's mistakes. FAB-SC addresses this by inducing two distinct verification behaviors. First, backsolve or plug-back checking conditions on the proposed answer and asks the model to enumerate constraints from the original question and check whether the answer satisfies them. This changes the reasoning direction and can expose inconsistencies that forward generation glosses over. Second, adversarial falsification explicitly asks the model to find a flaw, encouraging the production of negative evidence rather than elaboration of the original rationale.

Length and fluency as confounds in self-evaluation: Any verification score that is generated in natural language can correlate with surface-level properties such as verbosity. If longer rationales elicit higher confidence-like scores, weighting can inadvertently reward verbosity rather than correctness. FAB-SC therefore includes an explicit, deterministic penalty based on the token length of the forward rationale, computed outside the model, to reduce this confound while keeping the approach training-free.

4. Method

FAB-SC (Falsification-Augmented Bidirectional Self-Consistency) is a prompt-only inference and aggregation procedure that augments standard self-consistency with two complementary verification signals and a length-based regularizer, then selects an answer using importance-weighted voting. The method is designed for tasks with a single numeric final answer that can be extracted from model outputs.

Given a question q, FAB-SC begins with the same forward sampling stage as SC in spirit [1]. Using a CoT prompt template, it generates m chain-of-thought completions under stochastic decoding. From each completion it retains the rationale text r_i and parses a candidate numeric answer a_i. Samples for which the parser fails to extract a numeric answer are discarded.

For each retained sample i, FAB-SC computes three scalars that together represent the sample's estimated reliability.

Backsolve or plug-back verification (B-check): The method prompts the same base model with the original question q, the proposed numeric answer a_i, and the sampled rationale r_i as context. The prompt is structured to discourage free re-solving. Instead it instructs the model to (1) identify constraints and quantities from the question, (2) plug the proposed answer into these constraints, and (3) recompute implied quantities to confirm consistency. The response is required to include structured fields "CONFIDENCE: c" and "VERDICT: PASS or FAIL." FAB-SC parses the scalar confidence value, clamps it into [0, 1], and definesB$_iasthisclampedvalue$.

Adversarial falsification ensemble (F-check): For each forward sample i, FAB-SC runs k short stochastic "refuter" calls. Each refuter is given q and r_i and is instructed to search for particular error types such as quantity or unit misreads, arithmetic or algebra mistakes, or missing constraints. Each refuter outputs "ERROR$_PROB : p andaone - sentence_1REASON. FAB - SCparsesandclamps$pinto$[0, 1].TheensembleisaggregatedasF_i = 1 - (1/k) * sum_{t=1..k}p_{i,t}, sohigherF_imeansthesolutionwashardertorefute$.

Anti-verbosity regularization (length penalty): Let len$_i$ be the number of tokens in r_i as counted by the tokenizer in the model API wrapper. FAB-SC computes a deterministic penalty $L_i = \exp(-\lambda_{\text{length}} \cdot \text{len}_i / \text{tokens}_0)$, where λ_{length} is a nonnegative coefficient and tokens$_0$ is a reference scale. This makes L_i smaller for longer rationales.

Importance-weighted aggregation: FAB-SC combines the three signals into a single weight

$$w_i = B_i^{\alpha} \cdot F_i^{\beta} \cdot L_i,$$

where α and β are nonnegative exponents. Aggregation is performed at the answer level: for each distinct candidate answer value a, define $W(a) = \sum_{i:a_i=a} w_i$, and return the a with maximal $W(a)$. This can be viewed as a weighted generalization of SC's majority vote, where each sample contributes a vote scaled by estimated reliability rather than by 1.

Relative to SC, the central design choice is that agreement alone is not treated as sufficient evidence. B-check and F-check are intended to provide complementary signals that are less coupled to the forward solve trace than a single self-confidence score: constraint satisfaction evidence from backsolving and negative evidence from attempted refutation. The length penalty is included to reduce the chance that verification prompts overweight verbose rationales for superficial reasons. All components are inference-only and use the same base model, preserving SC's training-free constraint [1].

5. Experimental Setup

We evaluate FAB-SC against standard Self-Consistency (SC) in a controlled inference-only setting. Within each dataset, both methods use the same base model and the same forward sampling budget, so differences can be attributed to FAB-SC's additional verification and weighting rather than to increased forward sampling.

Datasets and splits: We use GSM8K (main configuration) and SVAMP, both standard benchmarks for arithmetic word problems in the CoT and SC literature [1,2]. For GSM8K, we evaluate on the test split restricted to the first 200 problems (pilot). For SVAMP, we evaluate on the test split restricted to the first 300 problems (pilot). Each dataset loader returns an identifier, the question string, and a numeric gold answer, converted to a float.

Model and interface: All runs use GPT-4o through the OpenAI chat completions API ($api_{m}odel_{n}ame$: gpt-4o-2024-08-06). The implementation uses a shared model wrapper that (i) issues generation calls, (ii) counts tokens for generated text, and (iii) supports n-way sampling for providers that expose it. Experiments are orchestrated using Hydra configuration files and logged to Weights and Biases.

Answer extraction and correctness criterion: For each model output, we extract a numeric answer using a rule-based function that searches for patterns including “The answer is X”, “Therefore, X”, “#### X”, or “Final answer: X”, and otherwise takes the last number in the output. Predictions and gold answers are stored as floats. An instance is marked correct if abs (predicted - gold) < 1e-6.

Baseline method: SC majority vote. For each question q, SC generates m forward chain-of-thought samples and returns the most frequent parsed answer. Both GSM8K and SVAMP baseline runs use $forward_{s}amples$ m = 8, temperature = 0.7, and $max_{t}okens$ = 512.

Proposed method: FAB-SC. FAB-SC uses the same forward sampling prompt and parameters as SC (m = 8, temperature = 0.7, $max_{t}okens$ = 512), then performs verification for each retained forward sample.

Backsolve verification details: Backsolve is enabled, with one verification call per forward sample. The verification generation uses temperature = 0.3 and $max_{t}okens$ = 512. The output is parsed for “CONFIDENCE” and “VERDICT.”

Falsification ensemble details: Falsification is enabled, with k = 3 refuter calls per forward sample. Each refuter uses temperature = 0.7 and $max_{t}okens$ = 256. The refuter prompts cycle among three focus statements (unit and quantity misreads; arithmetic or algebra mistakes; missing constraints) to encourage diversity.

Aggregation weight hyperparameters: alpha = 1.0, beta = 1.0, $\lambda_{\text{length}} = 0.001, andtokens0 = 200$.

Logging and artifacts: Each run saves per-question outputs to results.jsonl and writes aggregate metrics (accuracy, $correct_{c}ount$, $total_{c}ount$) to metrics.json. Weights and Biases logs per-question cumulative accuracy. A separate evaluation script can fetch run summaries and histories and generate cross-run comparison figures.

Evaluation metrics used in this paper: The primary metric is accuracy. Because the logged summaries also include wall-clock runtime, we report runtime as a coarse compute proxy. Other diagnostics proposed in the experimental design (for example, wrong-majority rescue rate, weight separability AUC, and length–weight correlation) are not part of the saved run summaries for the runs analyzed here and are therefore not reported.

6. Results

We report results from four pilot runs: SC and FAB-SC on GSM8K (first 200 test problems) and SC and FAB-SC on SVAMP (first 300 test problems). All runs use the same base model (GPT-4o) and the same forward-sampling budget (m = 8). The numbers below are taken from saved run summaries: accuracy, $correct_{c}ount$, $total_{c}ount$, and runtime.

Experiment 1: GSM8K pilot evaluation (N = 200) On GSM8K, the SC baseline achieves accuracy 0.810 (162/200). FAB-SC achieves accuracy 0.785 (157/200). Under the tested configuration, FAB-SC therefore underperforms SC by 0.025 absolute accuracy on this pilot slice.

The runtime gap is large. SC reports 1,164 seconds, while FAB-SC reports 20,587 seconds, an approximately 17.7× increase in wall-clock runtime. This overhead is consistent with FAB-SC issuing one backsolve call and three refuter calls per forward sample, in addition to the forward sampling calls.

Figure 1: Per-question cumulative accuracy trace for an accuracy-progress artifact produced by the evaluation pipeline; higher values are better. (filename: accuracy_progress.pdf)

Experiment 2: SVAMP pilot evaluation (N = 300) On SVAMP, the SC baseline achieves accuracy 0.8433 (253/300). FAB-SC achieves accuracy 0.8533 (256/300). Under the tested configuration, FAB-SC improves over SC by 0.010 absolute accuracy on this pilot slice.

Runtime overhead remains substantial. SC reports 1,548 seconds, while FAB-SC reports 27,764 seconds, an approximately 17.9× increase.

Cross-run accuracy comparison Across the four runs, the best baseline accuracy is 0.8433 (SC on SVAMP) and the best proposed accuracy is 0.8533 (FAB-SC on SVAMP), producing a best-of-set gap of +0.010. However, FAB-SC does not dominate SC across datasets: it improves on SVAMP but degrades on GSM8K.

Figure 2: Cross-run bar-chart comparison of final accuracies across runs; higher values are better. (filename: comparison_accuracy.pdf)

Figure 3: Cross-run overlay of cumulative accuracy progress across runs; higher values are better. (filename: comparison_accuracy_progress.pdf)

Compute-performance trade-off and fairness The comparisons are controlled within each dataset: SC and FAB-SC share the same base model and identical forward sampling parameters, so FAB-SC's extra runtime is attributable to its verification calls. Given the observed accuracy changes, the tested configuration does not establish a favorable compute-performance trade-off for FAB-SC. On GSM8K, the method is both slower and less accurate. On SVAMP, the method is much slower for a modest accuracy gain.

Limitations exposed by the available logs FAB-SC is motivated by wrong-majority behavior under correlated errors, but the available run summaries contain only aggregate accuracy and runtime. As a result, we do not directly measure how often FAB-SC flips an SC wrong majority, nor do we evaluate the separability of the internal weights w$_{i}$*forcorrectversusincorrectsamples.Similarly, thesavedsummariesdonotincludethedistributionsofrat*
to − endaccuracyandruntimeoutcomesonthetwopilotslices.

7. Discussion

8. Conclusions

We examined a practical weakness of Self-Consistency (SC): when multiple sampled chain-of-thought solutions share correlated reasoning errors, majority agreement can become a misleading proxy for correctness [1]. Under a strict prompt-only constraint, we proposed FAB-SC, which augments SC with two structured checks performed at inference time by the same base model: backsolve or plug-back verification that checks a proposed answer against problem constraints, and an ensemble of adversarial refuters that attempt to falsify the sampled solution. We combined these signals with a deterministic length-based penalty and used the resulting reliability weights for importance-weighted voting.

Pilot experiments with GPT-4o on arithmetic word problems showed mixed outcomes. On the first 200 GSM8K test items, FAB-SC underperformed SC (0.785 vs 0.810). On the first 300 SVAMP test items, FAB-SC slightly outperformed SC (0.853 vs 0.843). In both cases, FAB-SC increased wall-clock runtime by about 18× due to multiple verification calls per forward sample. These results suggest that prompt-induced verification can sometimes provide useful signal, but under our tested configuration it is not a consistently beneficial or compute-efficient replacement for majority-vote self-consistency.

The most direct next steps are to improve verification reliability and calibration (including more robust structured outputs and parsing), to evaluate FAB-SC with mechanism-targeted diagnostics such as wrong-majority rescue rates and weight separability, and to reduce verification cost via adaptive allocation and early stopping ideas that have been studied for lowering SC's inference budget [3].

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. Wang, X.; Wei, J.; Schuurmans, D.; Le, Q.; Chi, E.; Narang, S.; Chowdhery, A.; Zhou, D. Self-Consistency Improves Chain of Thought Reasoning in Language Models **2022**.
2. 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**.
3. Ma, M.; Gong, C.; Zeng, L.; Yang, Y.; Wu, L. Escape Sky-high Cost: Early-stopping Self-Consistency for Multi-step Reasoning **2024**.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.

https://doi.org/10.3390/1010000