Article

Cross-Perspective Verification Under Bounded Model Calls: Negative Results on GSM8K and SVAMP

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

Chain-of-Thought (CoT) prompting can improve multi-step reasoning, yet it remains brittle on numeric word problems because a single fluent rationale can silently entangle assumptions about equations, units, and constraints. This makes constraint violations hard to detect and encourages correlated errors when a model is asked to self-check in the same narrative style. We study CPV-CoT, a prompt-only, bounded-call procedure that enforces a representation change between solving and verification. CPV-CoT generates (1) a narrative step-by-step solution, (2) an independent equations-only re-solve, and (3) a short constraint-based adjudication only when the two candidate answers disagree, with an explicit abstention option. Using greedy decoding with Meta-Llama-3.1-8B-Instruct, we compare CPV-CoT against a single-pass zero-shot CoT baseline on 200-example test subsets of GSM8K and SVAMP. CPV-CoT increases cost to about 2.39–2.51 model calls per problem but reduces accuracy in these runs (GSM8K 0.625 vs 0.720; SVAMP 0.720 vs 0.805). We quantify disagreement-triggering frequency from logged call counts and discuss why cross-perspective prompting can fail under greedy decoding, clarifying what must be measured and improved for prompt-only verification to become reliably beneficial.

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) can become more accurate on multi-step problems when prompted to externalize intermediate reasoning. Chain-of-Thought (CoT) prompting formalized this phenomenon by demonstrating that eliciting natural-language rationales can substantially improve performance on arithmetic, commonsense, and symbolic reasoning tasks, including grade-school math word problems [**?**]. These gains, however, do not imply reliability. In practice, models can produce coherent explanations that contain subtle arithmetic slips, inconsistent variable bindings, or quiet violations of the original problem constraints, such as totals that do not sum correctly, counts that become non-integers, or units that drift mid-solution. For safety- or decision-critical settings, the core problem is not only whether an LLM can generate a plausible reasoning narrative, but whether it can detect when its own output violates the implicit contract defined by the question.

A key obstacle is representational. A single free-form rationale mixes many latent commitments: which quantities correspond to which symbols, which constraints are essential, which unit conversions are performed, and where rounding or simplification

Received:
Revised:
Accepted:
Published:

occurs. Because these commitments are encoded in prose, verification becomes difficult to operationalize. A common remedy is self-checking, asking the model to verify its previous solution. Yet if the verifier is prompted in the same narrative style and decoded greedily, solver and verifier can share failure modes. Informally, a model that made a mistake while telling one story is not necessarily well positioned to find that mistake by telling a second, similarly structured story.

This paper investigates a prompt-only alternative designed to introduce a deliberate representation change for verification while keeping inference cost bounded and auditable. The motivating analogy is a common human practice: solve in one way, then validate from a different angle, and spend additional effort primarily when the two perspectives disagree. We instantiate this as CPV-CoT (Cross-Perspective Verification Chain-of-Thought), a three-stage procedure that uses at most 3 model calls per problem. First, CPV-CoT produces a standard narrative step-by-step solution (Forward-CoT) with an explicit final numeric answer. Second, it independently re-solves the same problem in an equations-only style intended to compress reasoning into explicit algebraic and arithmetic transformations. Third, if the two candidate answers disagree (or one cannot be extracted), CPV-CoT triggers a short adjudication prompt that plugs candidate answers back into the problem constraints and outputs a discrete decision: choose answer A, choose answer B, or abstain (NONE).

The method is closely related to the CoT paradigm in that it elicits explicit intermediate reasoning, but it targets a more specific failure mode: the brittleness of verification when both solving and checking are performed in the same narrative representation. CoT work already highlights that prompting format affects both correctness and interpretability [**?**]. Our hypothesis is narrower and testable: verification should not merely be another narrative pass, but should shift representation (narrative versus equations-only) to reduce correlated solver-verifier errors. CPV-CoT also enforces a conditional compute budget: it always uses 2 calls (forward plus equations-only) and issues the third adjudication call only on disagreement.

We evaluate CPV-CoT on GSM8K and SVAMP, two benchmarks central to early CoT studies [**? ?**]. To isolate the effect of prompting structure rather than sampling variance, we use greedy decoding for all methods. The evaluation pipeline is fully specified in code: dataset preprocessing, prompt templates, answer extraction and normalization, and exact-match scoring. While the broader project description mentions an alternative model family, the recorded experiments reported here use Meta-Llama-3.1–8B-Instruct and conclusions are scoped to that setting.

Empirically, CPV-CoT does not improve end-to-end accuracy in our recorded runs. On 200 GSM8K test problems, a single-pass zero-shot CoT baseline attains 0.720 accuracy, while CPV-CoT attains 0.625. On 200 SVAMP test problems, the baseline attains 0.805 accuracy, while CPV-CoT attains 0.720. CPV-CoT also increases average calls per problem from 1.00 to approximately 2.39-2.51. At the same time, the control flow behaves as intended: disagreements between the two perspectives are common, triggering adjudication frequently.

Contributions:

- **Bounded cross-perspective procedure.** We formalize CPV-CoT, a prompt-only cross-perspective verification procedure combining a narrative solve, an equations-only re-solve, and a disagreement-triggered constraint adjudicator under a hard upper bound of 3 model calls.
- **Controlled evaluation pipeline.** We provide a fully specified evaluation pipeline (preprocessing, prompts, extraction and normalization, and scoring) and run controlled greedy-decoding comparisons on GSM8K and SVAMP with an 8B instruction model.

- **Negative results with call-count diagnostics.** We report and analyze negative results: CPV-CoT underperforms a single-pass zero-shot CoT baseline while using roughly 2.4-2.5 times as many model calls, and we quantify disagreement-triggering frequency from logged call counts.

These findings motivate future work that treats cross-perspective prompting as an empirical design space rather than an assumed improvement. In particular, follow-up studies should log disagreement and abstention behavior explicitly, diagnose when verification prompts degrade candidate quality, and examine how decoding strategies interact with bounded multi-step verification.

2. Related Work

2.1. Chain-of-Thought prompting for arithmetic reasoning

CoT prompting demonstrated that eliciting intermediate natural-language reasoning can improve performance on multi-step tasks, including arithmetic word problems, and that the effect becomes more pronounced with model scale [**?**]. Our baseline uses a simple zero-shot CoT instruction (solve step by step and provide a final number), closely matching the practical setting where no demonstrations are available. CPV-CoT shares the premise that intermediate reasoning is useful, but it differs in objective: it attempts to make verification less correlated with generation by forcing a change in representation between the first solution attempt and the verification attempt.

2.2. Automatic construction of CoT demonstrations

Auto-CoT reduces the manual effort of writing few-shot CoT demonstrations by clustering questions, selecting diverse exemplars, and using zero-shot CoT to generate rationales that become in-context demonstrations [**?**]. Conceptually, Auto-CoT improves the prompt prior through better demonstrations, whereas CPV-CoT allocates additional test-time compute per instance to produce multiple candidate answers and reconcile them. The approaches therefore differ in assumptions and where compute is spent. Auto-CoT typically assumes access to a pool of questions to cluster and build a demonstration set; CPV-CoT is entirely per-instance and requires no access to other questions at inference. While both are applicable to GSM8K and SVAMP, our experiments do not evaluate Auto-CoT because our goal is to isolate the effects of cross-perspective verification under greedy decoding rather than optimize in-context exemplars.

2.3. Prompt-only verification and conditional compute

CPV-CoT departs from common "self-checking" patterns primarily through (i) an explicit representation shift (narrative versus equations-only), and (ii) conditional compute, where adjudication is triggered only on internal disagreement. In principle, conditional compute can be important under tight inference budgets: verification is expensive, and spending additional calls uniformly on all problems may be wasteful. Our results underscore a complementary point: conditional compute can increase cost predictably, but it does not guarantee better accuracy. Additional calls also create additional failure opportunities (e.g., formatting and extraction failures, or an adjudicator selecting the wrong candidate), and the equations-only view can be systematically weaker than the narrative view under a fixed decoding regime.

2.4. Negative evidence relative to the CoT narrative

A central message of the original CoT literature is that producing explicit intermediate reasoning can improve end-task accuracy [**?**]. Our study highlights that adding more explicit reasoning steps is not automatically beneficial when those steps are composed into a multi-call pipeline and scored only by extracted final numbers. In our runs, CPV-CoT introduces new places where errors can occur, and the intended mechanism (representation change reduces correlated errors) is not realized as an end-to-end improvement. This motivates treating prompt-only verification procedures as systems whose components (solver, verifier, adjudicator, and extractor) must be evaluated jointly, under the same decoding and parsing constraints used in deployment.

3. Background

We study prompt-only methods for solving and verifying numeric word problems, focusing on settings where the final output must be a single numeric value that can be checked automatically. This section defines the task, the evaluation protocol used by our pipeline, and the prompting concepts required to interpret CPV-CoT.

3.1. Problem setting

A dataset consists of pairs (q, y), where q is a natural-language math word problem and y is the correct numeric answer. A method M maps q to a generated response r in free-form text. Because r may include prose and intermediate steps, evaluation uses an answer extraction function $E(r)$ that returns a normalized numeric string $\hat{y}$ (or an empty string if no answer can be extracted). The primary metric is accuracy, defined as the fraction of problems for which $\hat{y}$ matches y after normalization.

3.2. Answer extraction and normalization

Our evaluation implements deterministic extraction to reduce sensitivity to surface formatting. Given a model output, it first searches for an explicit marker of the form Final Answer: $\langle$number$\rangle$ (case-insensitive). If absent, it searches for a pattern like the answer is $\langle$number$\rangle$. If neither matches, it falls back to the last number token present anywhere in the output. Extracted numeric strings are normalized by removing commas and stripping whitespace; the code attempts to parse the result as a float and re-render it so that integer-valued floats become integer strings. Correctness is determined primarily by exact match of normalized strings, with an additional numeric comparison tolerance: if both predicted and gold strings can be parsed as floats, they are considered matching when the absolute difference is below 10^{-6}.

3.3. Chain-of-Thought prompting

CoT prompting elicits intermediate reasoning steps in natural language and has been shown to improve arithmetic reasoning, particularly for large models [**?**]. In our experiments, the baseline is a single-pass, zero-shot CoT instruction that requests step-by-step reasoning and requires the final answer in the explicit Final Answer: format to facilitate extraction.

3.4. Verification as a representation problem

The motivation for CPV-CoT is that a narrative rationale is not directly testable as a contract: it can describe constraints without forcing them into an operational form that is easy to re-check. CPV-CoT therefore introduces an equations-only representation for verification. The intent is to compress the same problem into explicit equations and arithmetic transformations while discouraging prose that could reproduce the same narrative mistakes.

3.5. Bounded compute and conditional verification

CPV-CoT bounds inference cost in model calls. Instead of using many sampled solutions or iterative refinement, it uses a small number of calls with a conditional third call executed only when internal disagreement is detected. This design makes cost predictable and allows some diagnostics to be inferred from call counts.

3.6. Datasets

GSM8K is a benchmark of grade-school math word problems whose answers are embedded in an explanation string and marked with #### ⟨number⟩, enabling reliable gold extraction [?]. SVAMP was designed to test robustness to linguistic variation in arithmetic word problems and has been used in CoT evaluations [? ?]. SVAMP examples store the prompt across two fields, Body (context) and Question (query). Our preprocessing concatenates Body and Question to form the full problem statement; this is treated as an essential assumption because using only one field yields incomplete problems and invalidates evaluation.

4. Method

CPV-CoT (Cross-Perspective Verification Chain-of-Thought) is a prompt-only inference procedure that generates two candidate answers using different response formats and, when needed, adjudicates disagreement by a constraint-focused check. It is implemented as deterministic control flow around standard text generation, using greedy decoding.

4.1. Control flow

Given a question q, CPV-CoT performs up to 3 model calls.

Algorithm 1 CPV-CoT inference with bounded calls

```
1: **Input:** question $q$
2: $f \leftarrow \text{GenerateForwardCoT}(q)$
3: $a \leftarrow E(f)$
4: $e \leftarrow \text{GenerateEquationSolve}(q)$
5: $b \leftarrow E(e)$
6: **if** $a \neq \emptyset$ **and** $b \neq \emptyset$ **and** $\text{Agree}(a, b)$ **then**
7:     **return** $a$
8: **else**
9:     $z \leftarrow \text{GenerateAdjudication}(q, a, b)$
10:    $c \leftarrow \text{ParseChoice}(z)$          $\triangleright\ c \in \{A, B, \text{NONE}\}$
11:    **if** $c = A$ **then**
12:        **return** $a$
13:    **else if** $c = B$ **then**
14:        **return** $b$
15:    **else**
16:        **return** $\emptyset$
17:    **end if**
18: **end if**
```

4.2. Stage prompts and decisions

Forward-CoT (narrative solve). The model is prompted: Solve this math problem using clear step-by-step reasoning. Explain your thought process in natural language. It is required to end with the explicit string Final Answer: ⟨number⟩. The returned text is stored as `forward_reasoning`, and a candidate numeric answer a is extracted using the deterministic extraction rules.

Equation-Solve (equations-only re-solve). The same question is re-issued under a representation constraint: Solve this math problem using ONLY mathematical equations and expressions. Do NOT use narrative prose—only write equations, calculations, and the final answer. It must again end with Final Answer: ⟨number⟩. The returned text is stored as `equation_reasoning`, and a second candidate answer b is extracted.

Disagreement adjudication (constraint plugging). If a and b agree, CPV-CoT accepts immediately. Agreement is checked first by exact string match of the normalized answers, and second by numeric comparison: if both parse as floats and $|a - b| < 10^{-6}$, they are treated as agreeing. If they do not agree (or if either answer is missing), CPV-CoT triggers an adjudication prompt that presents the original problem and both candidate answers and instructs the model to check which answer satisfies the problem constraints by plugging each candidate answer back into the problem's constraints. The adjudicator must output Choice: A, Choice: B, or Choice: NONE. The implementation maps these to decisions `ACCEPT_A`, `ACCEPT_B`, or `ABSTAIN`, and returns the corresponding final answer (or an empty string on abstention).

4.3. Bounded compute

Let N be the number of problems and D the number of disagreements that trigger adjudication. Forward-CoT and Equation-Solve always consume $2N$ calls. Adjudication consumes exactly D additional calls because it is executed at most once per problem. Therefore total_calls $= 2N + D$ and avg_calls_per_problem $= 2 + D/N$, with a hard upper bound of 3 calls per problem.

4.4. Implementation details from the code

CPV-CoT is implemented in a model wrapper that exposes `solve_zero_shot_cot` and `solve_cpv_cot`. Both use the same underlying `generate` function with greedy decoding (temperature 0.0 with `do_sample` disabled) and a maximum new-token budget of `max_new_tokens`= 512. The CPV-CoT function returns a structured record including the two intermediate outputs, the extracted candidate answers, the adjudication artifact when executed (prompt, output, decision), the final selected answer, and `num_calls`. This structure enables auditing of compute and supports post-hoc computation of diagnostics even when they are not logged during inference.

4.5. Intended mechanism and evaluation scope

CPV-CoT operationalizes the idea that verification should involve a representation change to reduce correlated solver-verifier errors. The disagreement signal between narrative and equations-only views is used as a trigger to spend additional compute. Our experiments test whether this mechanism translates into higher end-to-end accuracy when the full pipeline is run under greedy decoding and scored by regex-based answer extraction.

5. Experimental Setup

We evaluate CPV-CoT against a single-pass zero-shot CoT baseline on grade-school math word problems. The goal is to test whether a prompt-only verification procedure improves correctness under a bounded number of model calls, while holding decoding and answer extraction constant.

5.1. Datasets and sample sizes

We evaluate on GSM8K and SVAMP, both commonly used in CoT evaluations [**? ?**]. For each dataset, we run on a 200-example subset from the test split (`max_samples`= 200). GSM8K is loaded from the gsm8k dataset with the main configuration and test split. SVAMP is loaded from ChilleD/SVAMP using its test split.

5.2. Preprocessing and gold answers

GSM8K provides answers as explanation strings; preprocessing extracts the number following the #### ⟨number⟩ marker and normalizes it. SVAMP stores the problem statement in two fields, Body and Question. Preprocessing always concatenates them with a space to form the full question string and normalizes the gold numeric answer field.

5.3. Model and decoding

All runs use the instruction-tuned causal language model `meta-llama/Meta-Llama-3.1-8B-In` Decoding is greedy: temperature is 0.0, `do_sample` is disabled, and each generation uses `max_new_tokens`= 512. These settings are shared across baseline and proposed methods to isolate prompting effects.

5.4. Compared methods

The baseline method is Zero-shot CoT (single-pass): one model call with a prompt requesting step-by-step reasoning and requiring the final answer in the explicit format Final Answer: ⟨number⟩. The proposed method is CPV-CoT: two required calls (Forward-CoT and Equation-Solve) plus an optional adjudication call when candidate answers disagree.

5.5. Evaluation and logged metrics

For each example, we extract the final numeric answer using the regex-based policy described in Background and score correctness with exact match after normalization, plus a float tolerance of 10^{-6} when parsing succeeds. If CPV-CoT abstains (empty output) or produces no parseable number, it is scored as incorrect for the primary accuracy metric. The inference code logs run-level metrics: accuracy, correct_count, total_samples, total_calls, and avg_calls_per_problem. Although CPV-CoT motivates logging disagreement and abstention rates, these diagnostics are not present in the recorded run summaries. However, because adjudication adds exactly one model call per disagreement, the number of disagreements can be inferred as $D = \text{total\_calls} - 2N$.

5.6. Orchestration and run identifiers

Each method-dataset pair is executed via a Hydra configuration with `batch_size`= 1. The run IDs are `comparative-1-llama8b-gsm8k` (baseline on GSM8K), `proposed-llama8b-gsm8k` (CPV-CoT on GSM8K), `comparative-2-llama8b-svamp` (baseline on SVAMP), and `proposed-llama` (CPV-CoT on SVAMP). Runs are tracked with Weights and Biases, and results are saved to JSON files containing aggregate metrics and per-example records (question, ground truth, predicted answer, correctness, and method-specific artifacts).

6. Results

This section reports all recorded results for GSM8K and SVAMP (200 examples each), comparing a single-pass zero-shot CoT baseline to CPV-CoT. All numerical values are taken directly from the logged run metrics.

6.1. Experiment 1: GSM8K

The zero-shot CoT baseline attains accuracy 0.720 (144/200) with avg_calls_per_problem = 1.00 and total_calls = 200. CPV-CoT attains accuracy 0.625 (125/200) with avg_calls_per_problem = 2.51 and total_calls = 502. Thus, CPV-CoT both increases inference cost and reduces accuracy in this configuration.

Because CPV-CoT always executes exactly two calls per problem before optional adjudication, we can infer how often adjudication was triggered. With $N = 200$, the two mandatory stages cost $2N = 400$ calls. The logged total_calls is 502, so the excess is $D = 502 - 400 = 102$ adjudication calls. This implies a disagreement-trigger rate of $D/N = 102/200 = 0.51$ (51%). The frequent trigger confirms that the representation shift (narrative versus equations-only) produces substantially different candidate answers, but the downstream reconciliation does not improve end-to-end correctness under greedy decoding.

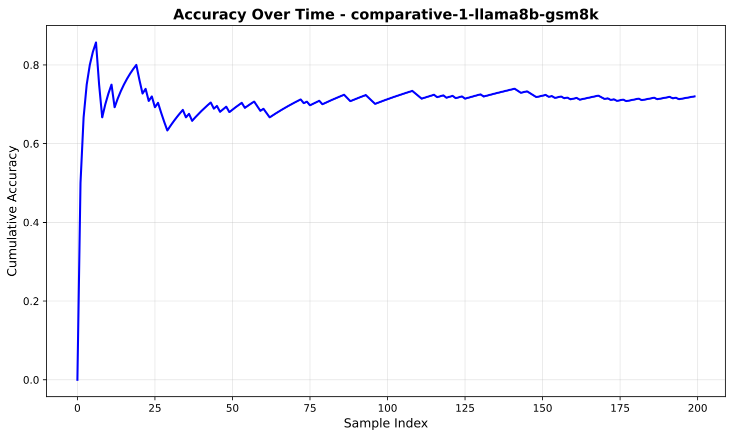

Figure 1. Cumulative accuracy over the first 200 GSM8K test examples for the zero-shot CoT baseline; higher values indicate better performance.

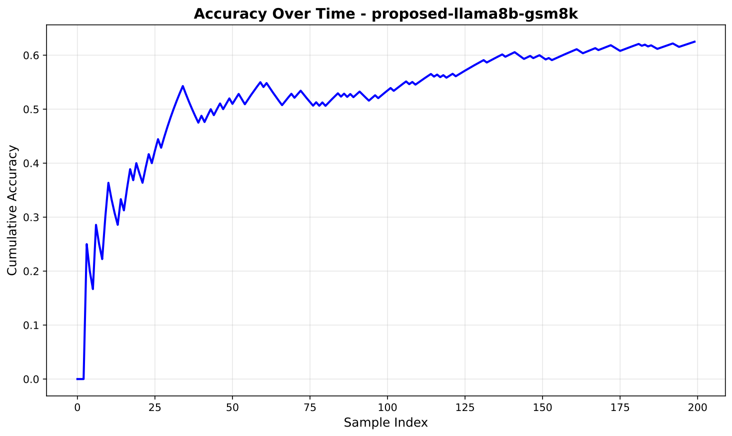

Figure 2. Cumulative accuracy over the first 200 GSM8K test examples for CPV-CoT; higher values indicate better performance.

6.2. Experiment 2: SVAMP

On SVAMP, the baseline attains accuracy 0.805 (161/200) with avg_calls_per_problem = 1.00 and total_calls = 200. CPV-CoT attains accuracy 0.720 (144/200) with avg_calls_per_problem = 2.385 and total_calls = 477. CPV-CoT is again more expensive and less accurate.

Inferring adjudication frequency, the mandatory two-stage cost is $2N = 400$ calls. With total_calls = 477, the excess is $D = 477 - 400 = 77$ adjudication calls, implying a disagreement-trigger rate of $77/200 = 0.385$ (38.5%). SVAMP triggers adjudication less often than GSM8K in these runs, but CPV-CoT still underperforms the single-pass baseline.

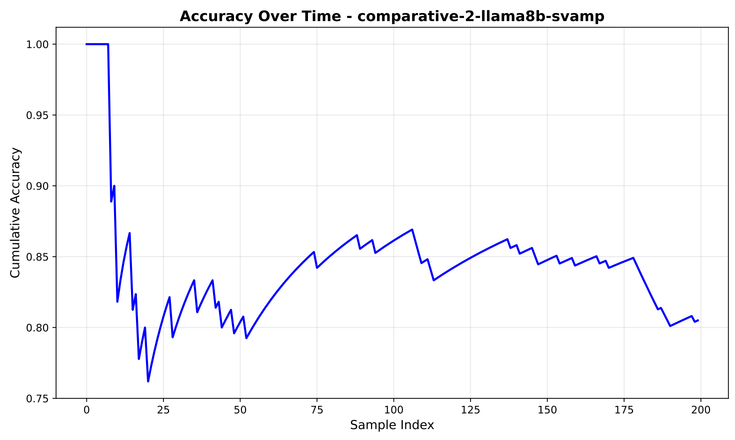

Figure 3. Cumulative accuracy over the first 200 SVAMP test examples for the zero-shot CoT baseline; higher values indicate better performance.

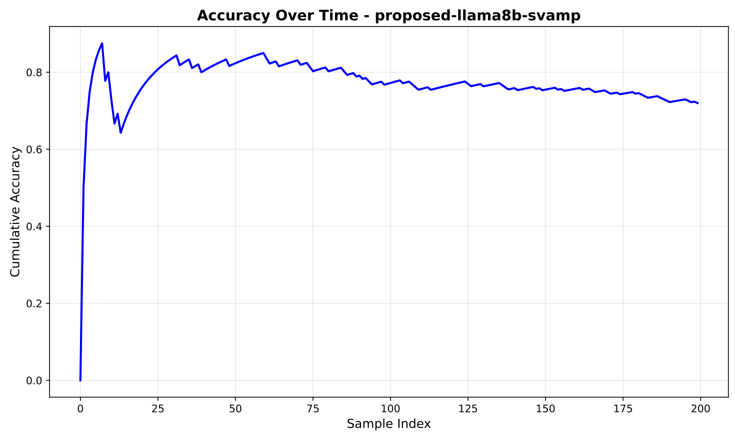

Figure 4. Cumulative accuracy over the first 200 SVAMP test examples for CPV-CoT; higher values indicate better performance.

6.3. Experiment 3: Cross-run comparison and compute trade-offs 301

For a summary across datasets, we compute macro-averages (simple mean across 302
GSM8K and SVAMP). Baseline macro-average accuracy is $(0.720 + 0.805)/2 = 0.7625$. 303
CPV-CoT macro-average accuracy is $(0.625 + 0.720)/2 = 0.6725$. For compute, the baseline 304
uses exactly 1.00 call per problem, while CPV-CoT uses $(2.51 + 2.385)/2 = 2.4475$ calls per 305
problem on average. The aggregated run comparison also reports best_baseline $= 0.805$ 306
and best_proposed $= 0.720$, reflecting that the highest accuracy among the recorded runs 307
is achieved by the baseline on SVAMP. 308

The evaluation script produced three additional summary plots. Figure 5 provides a 309
run-wise view of cumulative accuracy over examples as logged. Figure 6 summarizes final 310
accuracies by run. Figure 7 compares average calls per problem; for this cost metric, lower 311
values indicate better efficiency. 312

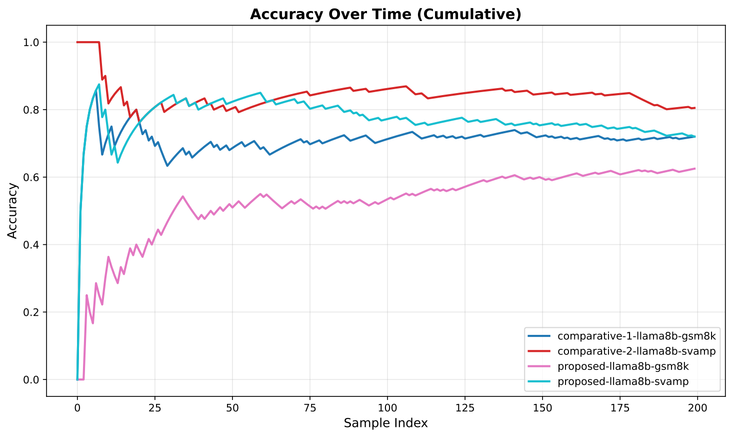

Figure 5. Accuracy over time comparison plot across runs as logged; higher values indicate better performance.

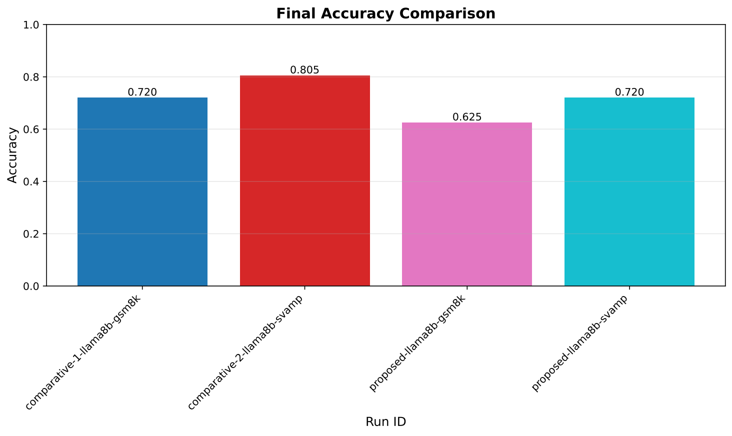

Figure 6. Final accuracy bar chart comparing runs; higher values indicate better performance.

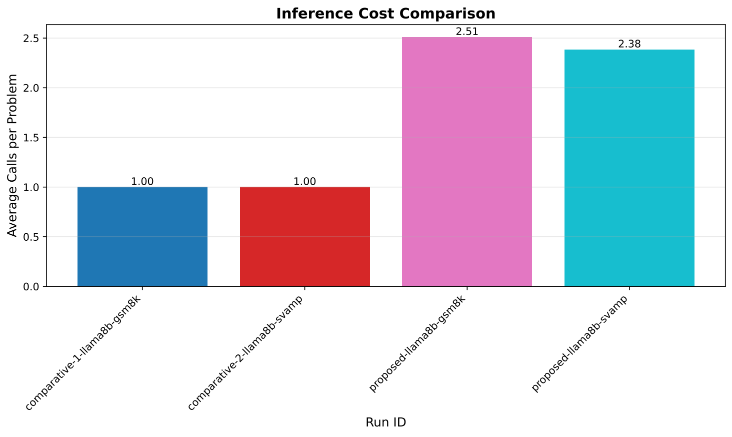

Figure 7. Average model calls per problem for each run; lower values indicate better performance for cost efficiency.

6.4. Limitations and fairness considerations 313

The comparison controls the model (Meta-Llama-3.1–8B-Instruct), decoding regime (greedy), token budget (512 new tokens), and answer extraction rules across methods, so the observed cost and accuracy differences are attributable to the prompting procedure and its additional calls. However, CPV-CoT introduces additional error channels that can plausibly explain degraded accuracy: each extra generation can produce an output that is difficult to parse into a single number under regex-based extraction; the equations-only view may be systematically weaker than the narrative view under greedy decoding; and the adjudicator is itself an unconstrained generation step that can select the wrong candidate or output abstentions.

Finally, CPV-CoT is motivated partly by the desire to reduce "confident wrong" outputs via an explicit abstention option. In these runs, `abstention_rate` and `wrong_non_abstain_rat` were not logged in the run summaries, so we cannot quantify whether CPV-CoT reduced non-abstaining errors even while lowering accuracy. This gap highlights the importance of aligning logged diagnostics with the intended safety motivation when evaluating prompt-only verification pipelines.

Overall, the recorded experiments demonstrate that cross-perspective verification is not automatically beneficial: in this evaluation setting, CPV-CoT yields lower accuracy at higher inference cost, despite frequently surfacing internal disagreement between its two perspectives.

7. Discussion

8. Conclusions

We studied CPV-CoT, a prompt-only cross-perspective verification procedure for math word problems that pairs a narrative Chain-of-Thought solution with an independent equations-only re-solve and applies a disagreement-triggered adjudicator that checks candidate answers against problem constraints. The method is motivated by a specific brittleness of single-pass CoT: a free-form narrative does not constitute an explicit, testable contract, and same-style self-checking can reproduce the same hidden assumptions and errors.

In controlled greedy-decoding experiments with Meta-Llama-3.1–8B-Instruct on 200-example test subsets of GSM8K and SVAMP, CPV-CoT increased compute to about 2.39–2.51 model calls per problem but reduced accuracy relative to a single-pass zero-shot CoT baseline (0.625 vs 0.720 on GSM8K; 0.720 vs 0.805 on SVAMP). Using call accounting implied by the method design, we inferred that disagreements were common (about 51% on GSM8K and 38.5% on SVAMP), confirming that the representation shift produces substantial divergence between the narrative and equations-only perspectives. Nevertheless, the adjudication step did not reliably convert this divergence into improved correctness under the present extraction and greedy decoding regime.

These negative results sharpen the empirical requirements for prompt-only verification. Representation change can surface inconsistencies, but improved end-to-end performance requires that the verifier and adjudicator be reliably complementary to the initial solve and robust to the same parsing constraints used for scoring. Future work should explicitly log disagreement outcomes and abstentions, refine prompts and output formats to reduce extraction brittleness, and examine how decoding choices interact with bounded multi-call verification.

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.