Article

CPV-CoT: Cross-Representation Self-Verification for Word-Problem Reasoning Under Bounded Model Calls

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 math word-problem solving, but its reliability remains limited: a single free-form rationale often entangles implicit assumptions about quantities, units, and constraints, making it hard for the model to detect when its own final number violates the problem statement. This is especially problematic when inference budgets rule out large self-consistency ensembles and when "confidently wrong" answers are costly. We study whether a prompt-only verification procedure can improve correctness by forcing a representation change between solving and checking while keeping computation bounded. We propose CPV-CoT, which (1) generates a narrative Forward-CoT solution, (2) independently re-solves the same problem in an equations-only format, and (3) conditionally invokes a constraint-focused adjudicator only when the two candidate answers disagree, with an explicit abstention option. We implement CPV-CoT and compare it to a single-pass zero-shot CoT baseline using meta-llama/Meta-Llama-3.1-8B-Instruct with greedy decoding on 200-example test subsets of GSM8K and SVAMP. Contrary to our hypothesis, CPV-CoT reduces accuracy while increasing average model calls by about 2.4–2.5 times. These negative results emphasize that representation diversity alone is insufficient for robust self-verification when the verifier is the same model and is not grounded by external execution.

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 be prompted to produce intermediate reasoning steps that improve performance on multi-step tasks, including grade-school arithmetic word problems [1]. In this setting, the model must translate narrative text into a structured set of quantities and relationships, perform several arithmetic operations, and preserve the constraints implied by the story (for example, non-negativity, integer counts, and totals that must match). Despite the apparent transparency of step-by-step explanations, CoT outputs frequently end in incorrect final numbers, often with high apparent confidence.

This paper starts from a specific hypothesis about why CoT remains brittle even when it looks coherent. A single free-form rationale tends to interleave multiple latent commitments—variable definitions, unit conversions, intermediate equations, and constraint assumptions—without making them explicit as a checkable "contract." When an error occurs, it is often unclear whether it is a local arithmetic slip, an earlier misread of the story,

Received:
Revised:
Accepted:
Published:

or a violation of a global constraint. Moreover, when the same model is asked to "verify" its answer, verification is typically elicited in the same narrative style as the original reasoning, making generator and verifier failures correlated.

A natural response is to use additional computation to reduce error: for example, sampling multiple solutions and selecting the majority answer. However, such approaches can be prohibitively expensive in budget-constrained deployments and are not always available for on-device or latency-sensitive use. We therefore ask a narrower question: can a prompt-only method, with a small and bounded number of model calls per problem, improve reliability by changing the representation used during verification?

We propose CPV-CoT (Cross-Perspective Verification Chain-of-Thought), a bounded-call procedure inspired by a human habit: solve the problem once, then validate it from a different angle. The key idea is that verification should not be "another pass of the same story-like CoT." Instead, CPV-CoT enforces two complementary perspectives: a narrative Forward-CoT solution and an independent equations-only solution that forbids prose. If the two extracted numeric answers agree, CPV-CoT accepts the answer immediately. If they disagree (or one is missing), CPV-CoT triggers a short adjudication step in which the model is instructed to plug candidate answers back into the original constraints and choose between candidate A, candidate B, or abstain.

To evaluate this hypothesis with minimal confounds, we adopt a controlled experimental setting: greedy decoding, fixed prompts, and small but reproducible test subsets. Using meta-llama/Meta-Llama-3.1-8B-Instruct, we compare a single-pass zero-shot CoT baseline to CPV-CoT on 200 test examples from GSM8K and 200 test examples from SVAMP. The results are negative. On GSM8K, the baseline achieves 0.720 accuracy (144/200), while CPV-CoT achieves 0.625 (125/200). On SVAMP, the baseline achieves 0.805 (161/200), while CPV-CoT achieves 0.720 (144/200). CPV-CoT also increases the average number of calls per problem from exactly 1.00 to 2.51 on GSM8K and 2.385 on SVAMP, implying frequent disagreements that triggered adjudication.

These findings matter for two reasons. First, they directly test a plausible and widely applicable intuition: that "multiple views" and verification can be created purely through prompting. Second, they clarify a limitation of prompt-only self-checking when the verifier is the same base model and no external execution is available: creating representational diversity can increase disagreement and compute without reliably improving the final decision.

Contributions: - We define CPV-CoT, a prompt-only, bounded-call inference procedure that combines a representation shift (narrative versus equations-only) with a disagreement-triggered adjudication step and an explicit abstention option. - We provide an end-to-end evaluation pipeline for GSM8K and SVAMP subsets, including dataset-specific preprocessing, answer extraction and normalization, and logging of inference cost. - We report negative results under greedy decoding with meta-llama/Meta-Llama-3.1-8B-Instruct: CPV-CoT reduces accuracy by 8.5–9.5 percentage points while increasing average calls per problem by about 2.4–2.5 times, and we quantify the implied disagreement frequency from call counts.

Future work should treat these results as diagnostic rather than final: the next step is to instrument the procedure with verification-specific metrics (disagreement rate by type, abstention rate, and wrong-non-abstain rate) and to explore adjudication mechanisms that are more directly grounded in constraint satisfaction than additional free-form generation.

2. Related Work

Our work is about prompt-induced reasoning and, more narrowly, about verification strategies for numeric word problems under strict inference budgets. We therefore focus on

prior work that studies how prompting elicits reasoning steps and how prompt construction affects robustness.

Chain-of-Thought prompting is the foundational technique. Wei et al. show that prompting LLMs to generate intermediate reasoning steps can substantially improve performance on arithmetic and symbolic reasoning tasks, including GSM8K and SVAMP, and that these benefits grow with model scale [1]. CPV-CoT operates within the same general paradigm (prompt-only inference) but targets a different failure mode: not whether the model can produce a coherent chain of thought, but whether it can detect and resolve constraint violations when its own reasoning is wrong. In this sense, CPV-CoT is closer to "verification prompting" than to elicitation alone.

Auto-CoT addresses a distinct source of brittleness: prompt sensitivity and the cost of manually crafting high-quality CoT demonstrations [2]. Auto-CoT automatically constructs a diverse set of demonstrations by clustering questions and generating rationales, then uses those demonstrations for single-pass inference. Compared with CPV-CoT, Auto-CoT spends extra model calls upfront to build in-context exemplars, whereas CPV-CoT spends extra calls at inference time for each query to obtain multiple candidate answers and adjudicate disagreements. This difference implies different cost profiles: Auto-CoT overhead can be amortized when demonstrations are reused, while CPV-CoT overhead scales linearly with the number of solved problems.

Our negative results provide a cautionary contrast to the general intuition that multi-view prompting necessarily improves reliability. Multi-view prompting is only helpful if (i) the alternative view is sufficiently accurate to produce informative disagreements and (ii) disagreements can be resolved by a dependable decision rule. In our setting, the adjudicator is the same base model and is not anchored by external computation, so it may rationalize incorrect candidates rather than reject them. Additionally, the forced "equations-only" format may itself be a performance bottleneck under greedy decoding, increasing disagreement frequency and thus compute.

Finally, a boundary of our comparison must be stated. The reference set provided here does not include dedicated verifier-model methods, execution-based checking, or sampling-based self-consistency approaches. Our experiments therefore compare CPV-CoT only to a single-pass zero-shot CoT baseline. The paper's contribution is not a broad state-of-the-art study, but a controlled test of whether a specific representation-shift verification heuristic improves outcomes under bounded calls, and an empirical demonstration that, under the tested conditions, it does not.

3. Background

We study numeric question answering for grade-school math word problems. Each example consists of a natural-language question q and a ground-truth numeric answer y. A method must generate text that contains a final numeric answer, and evaluation reduces that text to a canonical numeric representation for comparison.

Problem setting and notation. Let M denote an instruction-tuned LLM used as a conditional text generator. A prompting method is a bounded sequence of calls to M. For call i, the method constructs a prompt p
$_i$ *thatincludestheoriginalquestionqandmayincludeintermediateartifactsfromearliercalls.Themodelretu*
$_i = M(p$
$_i$*).AnanswerextractorEmapseacht*
$_i$*toacandidatenumericstringa*
$_i = E(t$
$_i$*);ifnoparseablenumberisfound,Ereturnsanemptystring.AnormalizerNmapsnumericstringsintoaca*

https://doi.org/10.3390/1010000

In our implementation, E uses a hierarchy of simple regex rules: (1) prefer an explicit marker of the form "Final Answer: <number>", (2) else search for a phrase like "answer is <number>", (3) else fall back to the last numeric token in the output. The normalizer N removes commas and attempts to parse a numeric string as a float; if the float is an integer, it is rendered without a decimal point. For agreement tests and correctness, we use exact string equality after normalization, with a fallback numeric comparison using absolute tolerance 1e-6 when both sides can be parsed as floats.

Baseline: single-pass zero-shot CoT. The baseline uses one model call with a prompt that requests step-by-step reasoning and instructs the model to end with the explicit "Final Answer: <number>" marker. This baseline is consistent with the general observation that eliciting intermediate reasoning can help multi-step arithmetic performance [1]. Importantly, it uses exactly one model call per problem.

Verification under bounded compute. The motivation for CPV-CoT is that many word-problem errors are not merely arithmetic slips; they involve misinterpreting a constraint in the narrative (for example, mixing totals and rates, miscounting items, or ignoring implicit non-negativity). When the solution is expressed as a single narrative, these constraints often remain implicit. Prompting the model to "verify" in the same narrative style may repeat the same hidden mistake. A principled way to reduce correlated generator–verifier errors is to force a representation change between the first solution and the check, while keeping the number of calls bounded.

This paper's setting therefore emphasizes two constraints: (i) prompt-only inference, without fine-tuning or external tools, and (ii) a small, fixed upper bound on the number of model calls per problem. In this setting, the core methodological question becomes whether representation diversity plus a deterministic accept-or-adjudicate rule can improve end-task accuracy.

4. Method

CPV-CoT (Cross-Perspective Verification Chain-of-Thought) is a prompt-only inference procedure defined by three prompts and a deterministic decision rule. It uses the same base model for all steps and does not rely on fine-tuning, external tools, or execution.

Given a question q, CPV-CoT produces two candidate solutions in complementary surface representations.

First call: Forward-CoT (narrative solve). The model is prompted to solve the problem using step-by-step natural-language reasoning and to end with "Final Answer: <number>". The output is t

$_fwdandtheextractedcandidateanswerisa$

$_fwd = E(t$

$_fwd)$.

Second call: Equation-Solve (compressed view). The model is prompted to solve the same question using only mathematical equations and expressions, explicitly forbidding narrative prose, and to end with "Final Answer: <number>". The output is t

$_eqnandtheextractedcandidateanswerisa$

$_eqn = E(t$

$_eqn).Thisforcedrepresentationshiftisintendedtochangehowthemodelexternalizesitsreasoningandthere$

Deterministic accept rule. If both a

$_fwdanda$

$_eqnarenon - emptyandN(a$

$_fwd)equalsN(a$

$_eqn)(eitherexactmatchafternormalizationornumericagreementwithinabsolutetolerance1e-6), CPV-$

$CoTreturnsthatsharedvalueandstops.$

Third call (conditional): Constraint adjudication. If the two candidates do not agree or one is missing, CPV-CoT triggers an adjudicator prompt. The adjudicator is shown the original question q and the two candidate numeric answers. It is instructed to check each candidate by plugging it back into the original constraints (units, counts, totals) and must output a discrete decision "Choice: A", "Choice: B", or "Choice: NONE". If the adjudicator chooses A or B, CPV-CoT returns the corresponding candidate value. If the adjudicator chooses NONE, CPV-CoT abstains.

Answer extraction and comparison details. Answer extraction and normalization follow the background definition of E and N. Within the model wrapper used for the experiments, numeric normalization attempts float parsing, removes commas, and collapses integer-valued floats (for example, "2.0" becomes "2"). The agreement test between candidates uses exact normalized-string match and then a numeric tolerance check when both are parseable floats.

Bounded call complexity and implied disagreement. CPV-CoT uses at most three model calls per problem. It always makes two calls (Forward-CoT and Equation-Solve) and makes the third call only when disagreement is detected. If d is the fraction of problems that trigger adjudication, then the expected number of calls per problem is 2 + d. In our experiments we log average calls per problem, so d can be inferred as (avg_calls_per_problem – 2).

Abstention and scoring. The prompt protocol includes an explicit abstention option (Choice: NONE). However, in the saved implementation the abstention decision is mapped to an empty predicted answer string, and the primary metric (accuracy) counts empty answers as incorrect. This design choice is appropriate for measuring correctness but can obscure potential safety-relevant benefits (for example, reducing wrong non-abstaining outputs). Consequently, in this paper we treat abstention as a method behavior but cannot quantify it because abstention-rate metrics were not logged in the saved run summaries.

5. Experimental Setup

We evaluate whether CPV-CoT improves the correctness of final numeric answers on arithmetic word problems, under greedy decoding and a bounded-call budget. The pipeline includes dataset preprocessing, prompt-based inference for each method, answer extraction and normalization, and metric logging.

Datasets and splits. We use two arithmetic word-problem benchmarks that are commonly used in CoT prompting studies [1,2]. For both datasets, we evaluate on the first 200 examples of the test split.

GSM8K: We use the gsm8k "main" test split. The ground-truth answers are stored as free-form text that includes a marker "#### <number>". Preprocessing extracts the number following "####", with a fallback to the last number in the answer text.

SVAMP: We use the ChilleD/SVAMP test split. SVAMP problem statements are stored in two fields, "Body" (context) and "Question" (the query). Preprocessing always concatenates Body and Question into a single question string to avoid incomplete prompts; this is treated as a correctness-critical preprocessing step.

Model and decoding. All reported runs use meta-llama/Meta-Llama-3.1-8B-Instruct, as specified in the run configuration files. Generation uses greedy decoding with temperature set to 0.0 (implemented as do
$_sample = False in the model wrapper). Each model call generates up to 512 new tokens.$

Methods compared.

Zero-shot CoT (baseline): A single model call prompts the model to solve step by step and to output "Final Answer: <number>". This method has a fixed inference cost of 1 call per problem.

CPV-CoT (proposed): Two calls are always executed (Forward-CoT and Equation-Solve). A third call (adjudicator) is executed only when the extracted candidates do not agree or when a candidate is missing.

Answer extraction, normalization, and correctness. Predicted answers are extracted from each model output by regex rules that prioritize the explicit final-answer marker and otherwise fall back to common phrases or the last numeric token. Predicted and ground-truth answers are normalized by removing commas and attempting float parsing. Correctness is determined by normalized equality, with a fallback numeric tolerance test using absolute error 1e-6.

Metrics and logging. During inference, the script logs cumulative accuracy after each sample index and stores final metrics including accuracy, correct_count, total_samples, total_calls, and average calls per problem (total_calls divided by total_samples). Although CPV-CoT is designed to expose disagreements and allow abstention, the saved run summaries do not include explicit disagreement rates, abstention rates, or adjudicator win rates. As a result, the quantitative evaluation in this paper focuses on the available metrics: accuracy and inference cost (calls per problem).

6. Results

We report results for four runs: zero-shot CoT and CPV-CoT on GSM8K (n = 200) and SVAMP (n = 200). All runs use meta-llama/Meta-Llama-3.1-8B-Instruct with greedy decoding (temperature 0.0) and a maximum of 512 generated tokens per model call.

Experiment 1: GSM8K accuracy and inference cost. The zero-shot CoT baseline achieves 0.720 accuracy (144/200) with exactly 1.00 call per problem (200 total calls). CPV-CoT achieves 0.625 accuracy (125/200) with 2.51 calls per problem on average (502 total calls). This is an absolute decrease of 0.095 accuracy while using 2.51 times as many model calls. Because CPV-CoT always performs two calls and adds one adjudicator call only on disagreement, the observed average implies an adjudication-trigger (disagreement) rate of approximately 2.51 – 2.00 = 0.51, or 51

For a simple post-hoc check of whether the observed decrease could plausibly be due to sampling noise at n = 200, we can compute an approximate two-proportion z test using the observed counts (144/200 versus 125/200). This yields a difference of –0.095 with an approximate standard error of 0.0467, giving z about –2.03 (p about 0.04). This calculation is only a rough diagnostic, but it suggests the drop is unlikely to be explained purely by random variation in the 200-item subset.

Figure 1: Cumulative accuracy over the first 200 GSM8K examples for the zero-shot CoT baseline run; higher values indicate better performance. (filename: comparative-1-llama8b-gsm8k
$_{a}ccuracy.pdf$)

Figure 2: Cumulative accuracy over the first 200 GSM8K examples for the CPV-CoT run; higher values indicate better performance. (filename: proposed-llama8b-gsm8k
$_{a}ccuracy.pdf$)

Experiment 2: SVAMP accuracy and inference cost. The zero-shot CoT baseline achieves 0.805 accuracy (161/200) with 1.00 call per problem (200 total calls). CPV-CoT achieves 0.720 accuracy (144/200) with 2.385 calls per problem (477 total calls). This is an absolute decrease of 0.085 accuracy while using 2.385 times as many calls. The implied adjudication-trigger (disagreement) rate is approximately 2.385 – 2.00 = 0.385, or 38.5

A similar post-hoc two-proportion calculation on SVAMP (161/200 versus 144/200) gives a difference of –0.085 with an approximate standard error of 0.0424, yielding z about –2.00 (p about 0.045). Again, this is not a preregistered statistical test, but it supports the qualitative conclusion that the regression is sizeable relative to the subset size.

Figure 3: Cumulative accuracy over the first 200 SVAMP examples for the zero-shot CoT baseline run; higher values indicate better performance. (filename: comparative-2-llama8b-svamp
$_{a}ccuracy.pdf$)

Figure 4: Cumulative accuracy over the first 200 SVAMP examples for the CPV-CoT run; higher values indicate better performance. (filename: proposed-llama8b-svamp
$_{a}ccuracy.pdf$)

Experiment 3: Cross-run comparison of quality and cost. The aggregated comparison plots summarize (i) cumulative accuracy trajectories across runs, (ii) final accuracy per run, and (iii) average calls per problem. Under the reported settings, CPV-CoT underperforms the baseline on both datasets and simultaneously consumes substantially more inference compute, so it is Pareto-dominated by single-pass zero-shot CoT.

Figure 5: Cumulative accuracy comparison across all runs; higher values indicate better performance. (filename: comparison
$_{a}ccuracy.pdf$)

Figure 6: Final accuracy comparison across all runs; higher values indicate better performance. (filename: comparison
$_{a}ccuracy$
$_{b}ar.pdf$)

Figure 7: Average model calls per problem across runs as an inference-cost metric; lower values indicate better performance. (filename: comparison
$_{c}alls.pdf$)

Limitations and interpretability of the negative result. The available metrics establish that CPV-CoT reduces end-task accuracy and increases cost through frequent adjudication triggers. However, the saved run summaries do not provide the verification-specific diagnostics needed to localize the failure mode. In particular, we do not have: (i) explicit disagreement rates defined at the candidate-answer level (beyond what can be inferred from average calls), (ii) abstention frequency (Choice: NONE) and its contribution to the accuracy drop, or (iii) adjudicator win rates, meaning how often adjudication selected the correct candidate when one of the two candidates was correct. Without these measurements, we cannot determine whether the regression is driven primarily by a weak equations-only solve, an unreliable adjudicator, the interaction of abstention with the accuracy metric, or a combination of these factors under greedy decoding.

7. Discussion

8. Conclusions

We investigated whether prompt-only verification can make Chain-of-Thought prompting more reliable for arithmetic word problems under a tight, bounded-call budget. CPV-CoT enforces a cross-representation check by generating two candidate answers from the same model (a narrative Forward-CoT and an equations-only solve) and invoking a constraint-focused adjudicator only when the two candidates disagree, with an explicit abstention option.

On 200-example test subsets of GSM8K and SVAMP using meta-llama/Meta-Llama-3.1-8B-Instruct with greedy decoding, CPV-CoT does not improve accuracy. Instead, accuracy drops from 0.720 to 0.625 on GSM8K and from 0.805 to 0.720 on SVAMP, while average calls per problem increase from 1.00 to 2.51 and 2.385, respectively. The call counts imply frequent disagreements approximately 51% on GSM8K and 38.5% on SVAMP, meaning the method often pays for adjudication but does not convert that extra compute into better final answers.

These negative results refine the broader lesson that verification is not simply "more reasoning." When the checker is the same base model and verification is not grounded by external execution, representation diversity can expose inconsistencies but still fail to resolve them correctly. Future work should add the missing diagnostics (explicit disagreements, abstention rates, and wrong-non-abstain rates) and explore decision rules or verification signals that make constraint satisfaction more mechanically testable rather than relying on additional free-form text generation.

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**.
2. Zhang, Z.; Zhang, A.; Li, M.; Smola, A. Automatic Chain of Thought Prompting in Large Language Models **2022**.