ARE TWO MINDS BETTER THAN ONE? EVALUATING COLLABORATIVE REASONING IN LLMS

Daria Stănescu, Maria Chiper, Traian Becheru, Laura Ruse, Dinu Țurcanu

Large language models (LLMs) have shown strong capabilities in reasoning and problem solving, but their outputs often suffer from inconsistency, shallow verification, and occasional errors. Recent research has suggested that collaboration between multiple LLMs, inspired by human group problem solving, could mitigate these weaknesses. This paper investigates whether collaborative interactions between large language models (LLMs) can outperform solo reasoning. We compared single-agent reasoning against two-agent collaborations using two modes: debate and consensus. Using a quota-aware evaluation threshold, we tested GPT-4.1 and GPT-4o on 30 tasks across two categories: (1) problem solving (mathematical, logical, and role-based reasoning), and (2) fact-checking and factual synthesis. Our results suggest that collaboration improves robustness in tasks requiring multi-step reasoning and factual cross-validation, but it introduces additional latency and token usage. We discuss trade-offs between accuracy, efficiency, and cost, providing evidence-based guidance on when "two minds are better than one" in LLM reasoning.

Keywords: large language models; collaboration; multi-agent systems; debate; consensus; reasoning evaluation, automatic scoring

1. Introduction

Large language models (LLMs) have achieved impressive progress in recent years [1], showing strong capabilities in problem solving, reasoning, and factual recall. However, despite these advances, their outputs are still prone to errors and hallucinations, particularly when tasks require multiple reasoning steps or factual verification. A natural question arises: is collaboration

Faculty of Automatic Control and Computers, National University of Science and Technology Politehnica Bucharest, Romania, e-mail: daria.stanescu@stud.acs.upb.ro

University of Bucharest, Romania, email: maria.chiper1@s.unibuc.ro

Faculty of Automatic Control and Computers, National University of Science and Technology Politehnica Bucharest, Romania, e-mail: traian.becheru@stud.acs.upb.ro

Faculty of Automatic Control and Computers, National University of Science and Technology Politehnica Bucharest, Romania, e-mail: laura.ruse@upb.ro

Faculty of Electronics and Telecommunications and National Institute of Innovations in Cybersecurity "CYBERCOR", Technical University of Moldova, Moldova, e-mail: dinu.turcanu@adm.utm.md

between two models better than solo reasoning, in the same way that human group interactions often outperform individual problem solving?

Previous studies have explored collaborative setups for LLMs, introducing debate-style frameworks [2, 3] and consensus-based approaches [4, 5]. These works suggest that collaboration can improve reasoning accuracy and fact-checking, but they also highlight limitations such as increased token consumption, redundancy, and the risk of amplifying hallucinations or training biases [6]. The scientific problem remains: when is collaboration between LLMs genuinely beneficial, and when does it become inefficient?

This study aims to provide an evaluation of collaborative LLM reasoning. We directly compare solo versus collaborative agents using a homogeneous approach (meaning we will be using the same LLM model for both agents when testing collaborative solutions). While building our automated testing strategy, we have decided to follow two methods of collaboration between the agents:

- **Debate mode**: Two agents independently produce solutions and a "judge", similar to Liang's et al. approach in [2], synthesizes the best outcome taking into account the pros and cons of both solutions.
- **Consensus mode**: Agents assume complementary roles (researcher and verifier), and an aggregator reconciles their answers.

Our results show that collaboration improves accuracy by approximately 10% compared to solo reasoning. The gains are particularly strong in multi-step problem solving, where both debate and consensus reduce arithmetic and logical errors. However, these improvements come at a cost: collaborative modes require roughly three to four times more wall-clock time and tokens, which lowers efficiency. Interestingly, debate tends to perform better in fact-checking tasks, while consensus is most effective in structured problem solving.

The remainder of this article is structured as follows. Section 2 reviews related work on collaborative LLMs and collective intelligence. Section 3 describes our experimental setup. Section 4 presents the results, including overall accuracy, efficiency, and error analysis. Section 5 discusses the trade-offs between our defined collaboration modes and when solo reasoning is more appropriate. Finally, Section 6 presents our conclusions and directions for future research.

2. Related Work

Research into enhancing LLM reasoning has largely evolved from prompt engineering techniques for single models to complex multi-agent collaborative systems.

2.1. Single-Agent Reasoning Strategies

Prior work shows that making intermediate reasoning explicit can help large language models (LLMs) solve multi-step problems more reliably. Chain-of-Thought prompting encourages models to generate reasoning traces before the final answer [7], significantly improving performance on math and logic tasks. Self-Consistency extends this by sampling multiple reasoning paths and aggregating results via majority vote [8], effectively marginalizing out random generation errors. Tree-of-Thoughts further structures this process by exploring multiple partial plans in a search tree [9].

However, single-agent methods remain vulnerable to the "hallucination snowball" effect: once a model commits to an incorrect intermediate step, it often doubles down on the error to maintain internal consistency, not having an external mechanism to trigger any correction. Benchmarks like TruthfulQA highlight this distinct failure mode, measuring the tendency of models to produce confident, misleading answers that "look right" [10]. The persistence of these errors motivates our specific inclusion of a fact-checking category and a judge/aggregator that must explicitly justify its choices to break this cycle.

2.2. Multi-Agent Collaboration and Debate

To address single-agent limitations, recent literature has shifted toward multi-agent systems that leverage social behaviour. Du et al. [11] and Liang et al. [2] independently demonstrated that multi-agent debate allows models to critique one another, exposing contradictions that a single isolated model might overlook. This "divergent thinking" mitigates confirmation bias and improves factual accuracy.

Beyond simple debate, frameworks like *AutoGen* [3] introduced flexible infrastructures for role-playing agents, allowing specialized personas (e.g., Coder vs. User) to solve tasks that stall single agents. More recently, Feng et al. [5] utilized multi-LLM collaboration to identify knowledge gaps, allowing the system to abstain rather than hallucinate when the consensus is low. Similarly, Duan and Wang [4] explored integrating third-party LLMs to enhance consensus and analyze uncertainty.

2.3. Reliability, Evaluation, and Broader Contexts of LLM Collaboration

Recent work also suggests that the value of multi-agent or multi-model interaction should be understood not only in terms of final-answer accuracy, but also in terms of agreement, measurement stability, and the ability of models to make their own judgments more explicit. In studies of AI-assisted content analysis, Rughiniș et al. [12], [13] show that LLMs can reach substantial agreement when they are asked to code complex qualitative material, yet their consistency remains sensitive to prompt design, task framing, and the interpretive ambiguity of the target phenomenon. This perspective complements

debate-based reasoning frameworks by emphasizing that collaboration is useful not only because agents can challenge each other, but also because disagreement itself can be treated as a measurable signal of uncertainty.

A similar concern with consistency appears in adjacent work on inter-rater reliability and AI-supported classification. Ioan et al. [14] examine LLM consistency in coding political debates, while Rughiniș et al. [15] discuss how generative AI is increasingly positioned as a hybrid actor in academic knowledge production, where questions of trust, accountability, and human oversight become central. These studies broaden the interpretation of collaborative LLM systems: they are not only computational strategies for improving reasoning, but also socio-technical arrangements in which reliability must be evaluated at the level of procedures, roles, and governance rules.

This broader evaluative perspective is also visible in application-oriented studies. For example, Ghiță et al. [16] assess chatbot performance in cybersecurity challenges, showing that structured benchmarking can reveal where LLMs are dependable and where domain-specific limitations persist. Such findings are relevant for our study because they support a shift from asking whether collaboration is universally better than solo prompting to asking under what conditions collaboration produces robust, verifiable, and cost-effective improvements. Our experiments follow this line of inquiry by comparing homogeneous collaborative setups against a solo baseline while explicitly tracking both accuracy and efficiency.

Some broader context can also be drawn from other recent works on related topics. Badea et al. [17] analyze convergence and performance trade-offs in asynchronous federated learning, highlighting how distributed collaboration benefits from explicit attention to heterogeneity and system coordination. Sandescu et al. [18] show, in a cybersecurity setting, that natural language processing pipelines can support the structured extraction of exploits and attack vectors from unstructured news reports, which is relevant to our fact-checking and factual synthesis perspective. In turn, Contasel et al. [19] emphasize resilience and modular verification in complex AI-enabled infrastructures, reinforcing the idea that collaborative intelligence should be evaluated not only by outcome quality, but also by architectural robustness, interpretability, and operational cost.

2.4. **Limitations and The Efficiency Gap**

Despite these advancements, significant gaps remain regarding the efficiency and experimental isolation of these methods.

First, existing studies often rely on heterogeneous pools of models (e.g., GPT-4 debating Llama-2). While effective, this introduces confounding variables: it is difficult to determine whether the improvement is due to the collaboration strategy or simply the superior knowledge of the stronger model. Furthermore, theoretical concerns exist regarding homogeneous systems. Bao

et al. [6] note that homogeneity in LLM interactions can lead to amplification bias, suggesting that identical models might simply reinforce each other's errors rather than correct them.

Second, there is a lack of direct analysis on the cost-quality trade-off. Multi-agent loops often require multiple rounds of generation, tripling or quadrupling token usage. Prior work rarely analyzes whether the marginal gain in accuracy justifies the exponential increase in inference costs.

Our work addresses this gap by strictly controlling the model type (homogeneous) and quantifying efficiency in relation to cost (accuracy per token) alongside the performance gain. By leveraging strategies to counter known model weaknesses, we investigate whether identical agents can overcome amplification bias through structured collaboration, ensuring that performance gains are driven by the collaborative framework itself rather than the capabilities of a superior model.

3. Solution Design

To evaluate the impact of collaboration on reasoning accuracy in a homogeneous system, we designed a comparative framework consisting of three distinct interaction approaches. We tested a standard single-agent baseline against two multi-agent setups: one focused on Debate (adversarial critique) and one on Consensus (cooperative verification). Crucially, we used the exact same AI model for every role in each setup. This ensures that any improvement in performance is due to the collaboration method itself, rather than the capabilities of a superior model.

We implemented these setups as structured prompt templates, defined as follows:

CM0: Solo (1 call/task): A single `user` message with the task; the model returns one answer. The model responds directly without any role separation or collaboration, following the classic single-turn prompt–completion paradigm that is standard in most LLM applications.

CM1: Debate (3 calls/task): Two independent openings: *Planner* produces a plan, and *Critic* challenges it by executing the solution and flagging potential flaws. A third "Judge" prompt arbitrates between them. This setup was chosen because prior research in collective intelligence [11] shows that adversarial collaboration can surface hidden errors and improve fact-checking. However, to remain within API quota budgets, we restricted the debate to avoid running extended back-and-forth dialogue.

```
 [...](
        "ROLE: Planner.\n"
         [...]
        "OUTPUT: Plan with numbered steps,
         key assumptions, hardest steps."
    )}
```

```
[...](
       "ROLE: Critic.\n"
        [...]
       "OUTPUT: Attempted solution with step numbers;
        state disagreements with the plan."
   )}
[...](
       "You are the judge. Two answers are below.\n"
       [...]
       "Your task is to critically evaluate both answers.
       Identify flaws, unsupported claims, or reasoning
       errors, then decide which answer is stronger.
       Output JSON with keys: final_answer, reasons."
   )}
```

CM2: Consensus (3 calls/task).: Two complementary roles: *Researcher* outputs `steps, answer, confidence`; *Verifier* outputs `checks, answer, confidence`. A third Aggregator synthesizes both into `final_answer, why`. The confidence parmeter, inspired by [20] is used to help distinguish between real consensus and shallow agreement and to help avoid confirmation bias.
The complementary role outputs are encouraged to be JSON-like to simplify aggregation.

```
[...](
       "ROLE: Researcher.\n"
       [...]
       "REQUIREMENTS: produce steps, answer, confidence (0{100). "
       "OUTPUT JSON keys only: steps (list), answer (string),
        confidence (number)."
   )}

[...](
       "ROLE: Verifier.\n"
       [...]
       "REQUIREMENTS: execute, check steps, list potential
        errors, answer, confidence (0{100). "
       "OUTPUT JSON keys only: checks (list), answer (string),
        confidence (number)."
   )}

[...](
       "You are the aggregator. You will see two JSON solutions:\n\n"
       [...]
```

```
            "Your task is to combine their complementary strengths.
             If the answers agree, adopt the shared solution.
             Output JSON with keys: final_answer, why."
        )}
```

All collaborative prompts include a shared `system` message enforcing role fidelity and consistent behavior:

```
SYSTEM_SHARED = (
    "You are part of a two-expert collaboration. Follow your role strictly."
    "Be concise, justify non-trivial steps, and obey any requested output
    schema."
    "Do not mention model identities."
)
```

4. Experimental Setup

4.1. Models

We evaluated two large language models (LLMs) exposed through the GitHub Models API[1] (OpenAI-compatible schema) :

- **GPT-4.1** (`openai/gpt-4.1`): a high-performance reasoning model.
- **GPT-4o** (`openai/gpt-4o`): optimized for balanced performance and efficiency.

All calls were executed using the official OpenAI Python SDK [2]. Decoding parameters were `temperature = 0.65` and `max_tokens = 300`, in order to limit the amount of tokens used per answer while still keeping a relatively open boundary for responses. This setting aims to approach a human-like behavior, hence the temperature used represents a balance between determinism and creativity, allowing the models to explore alternative reasoning paths. Each request used a simple exponential backoff with at most three retries (`RETRY_MAX = 3`, base sleep = 1.5 s) to mitigate transient gateway errors.

4.2. Tasks and Prompt Sets

We constructed two evaluation categories totaling 30 prompts:

- **Problem Solving** (15 prompts): mathematical problems, arithmetic with units and logic puzzles. For grounding, we drew inspiration from the GSM8K dataset [21], introduced by Suzgun et al. at Google and the MATH dataset [22], proposed by Hendrycks et al. which are widely used benchmarks for evaluating arithmetic reasoning and step-by-step problem solving in LLMs.

[1] GitHub Models: `https://docs.github.com/en/rest/models/catalog?apiVersion=2022-11-28`

[2] OpenAI Python SDK: `https://platform.openai.com/docs/libraries/python-library`

- **fact-checking and Factual Synthesis** (15 prompts): verification of claims and comparing or contrasting information from different inputs. These prompts often require citing reliable facts or checking consistency across multiple statements. We based this category on the FEVER dataset [23], created by Thorne et al., the TruthfulQA benchmark [10], which measures the tendency of models to produce factually accurate versus misleading answers, and the SciFact dataset [24], which focuses on scientific claim verification. FEVER remains one of the most established resources for testing factual accuracy.

Each prompt in the CSV includes: a unique ID, category, the task text, the right answer, and an evaluation type (`contains`, `regex`, `bool`, or `number`). To respect API budgets, limiting the number of samples used from the prompt file can be done via `--limit` argument when executing the script.

4.3. Experimental Matrix

We ran **homogeneous** experiments (the same model is used for both agent roles in collaborative modes) across three conditions per model (Table 1). Each experiment targeted up to $n = 30$ prompts per condition, subject to rate limits.

Table 1. Experimental grid. Each condition targeted up to 30 prompts, subject to API quotas.

Experiment	Model	No. of Agents	Mode	No. of Experiments
E1	GPT-4.1	1	CM0	30
E2	GPT-4.1	2	CM1	30
E3	GPT-4.1	2	CM2	30
E4	GPT-4o	1	CM0	30
E5	GPT-4o	2	CM1	30
E6	GPT-4o	2	CM2	30

To make cost comparisons explicit, we report the number of API calls per task (Table 2). Debate and Consensus each require three calls per task (two role calls + one aggregation/judge call).

Table 2. Calls per task by collaboration mode.

Mode	Agent Structure	Calls per Task
CM0	1 agent, direct answer	1
CM1	2 openings + judge	3
CM2	2 openings + aggregator	3

4.4. Python-Based Script Pipeline Architecture

All experiments were executed with a Python script which provides:

(1) **Model Invocation.** Requests were sent through the GitHub Models API [3] using the official OpenAI Python SDK . Each call was wrapped in a retry loop (`RETRY_MAX = 3`) with exponential backoff (base sleep = 1.5 s) to mitigate transient gateway errors. The script records end-to-end latency for every request

(2) **Prompt Management.** Experimental tasks were defined in `prompts.csv` and loaded into structured objects (`PromptItem`). A command-line flag ($--limit$) allowed restricting the number of prompts for rapid iterations and respecting the quota limit.

(3) **Collaboration Modes.** Three collaboration modes were implemented, as stated before: CM0 (Solo), CM1 (Debate), and CM2 (Consensus). Each mode constructs role-specific prompt templates and performs the required number of API calls (1 for Solo, 3 for Debate and Consensus). The final output is produced by either the "Judge" (Debate) or the "Aggregator" (Consensus).

(4) **Automatic Scoring.** Generated answers were scored against correct solutions, included in `prompts.csv`, using one of five evaluation types: `exact`, `contains`, `regex`, `bool`, and `number`. Each evaluation type is self-explanatory and allowed consistent evaluation of factuality and correctness of answers.

(5) **Metrics.** For each task, the script logged correctness, latency, and token usage (prompt + completion). Since usage statistics are not always returned by the API proxy, a fallback estimator was used (*tokens* $\approx 1.3\times$ word count). Results are saved both per experiment (`runs_E*.csv`) and in a consolidated `summary.csv` file.

4.5. Outcome Measures

For each experiment (E1–E6), the results aggregate:

- **Accuracy**: mean of binary correctness across prompts.
- **Latency**: mean total wall-clock seconds per completed task (summing role calls for collaborative modes).
- **Tokens**: mean estimated total tokens (prompt + completion; summing role calls).
- **Efficiency**: accuracy per 100 tokens, computed as

$$\text{Eff.} = \frac{\text{Accuracy} \times 100}{\text{Avg. Tokens}}.$$

[3] GitHub Models API inference endpoint: `https://models.github.ai/inference`

4.6. Quality Control and Failure Handling

To keep the experiments robust and reproducible, the script incorporates an automatic retry mechanism for requests that fail due to temporary or transient API errors, such as HTTP 429 (rate limit exceeded) or 5xx server-side responses. Rather than immediately aborting on failure, the script waits for a short configurable delay between attempts, giving the API time to recover before retrying. This approach prevents isolated network hiccups or momentary service degradation from prematurely terminating a long-running experiment. If a request continues to fail after reaching the maximum number of allowed retries, the failure is not silently discarded. Instead, it is explicitly recorded as an `[ERROR]` entry in the output, ensuring that the CSV files remain structurally complete and that every task slot is accounted for. This design choice is deliberate: silent data loss could introduce subtle biases into downstream analysis, making it appear as though certain conditions performed better or worse than they actually did. By preserving a visible error marker, the researcher is always aware of which entries could not be completed and can decide how to handle them during post-processing. In collaborative or multi-agent settings, where a single task involves multiple model roles interacting in sequence, the failure handling is stricter. If any one of the role prompts fails, whether due to an API error or an unexpected response format, the entire task is marked as failed rather than continuing with partial results. This prevents a degraded or incomplete interaction from being passed to the judge or aggregator, which could otherwise introduce misleading scores into the evaluation. Finally, to make the execution process transparent and easy to monitor, the script emits step-by-step logs throughout its run, indicating details such as which role is currently being executed, which task is in progress, and when retries are being attempted. These logs allow the researcher to follow the experiment in real time and quickly identify any recurring issues without needing to inspect the output files directly.

5. Results

5.1. Overall Performance

Table 3 summarizes overall performance (both models, all prompts). Collaboration improved accuracy by +10 points on average versus Solo, with the largest gains observed in multi-step problem solving. However, collaborative modes required roughly 3–4× more wall-clock time and 3× more tokens per task, reducing efficiency (accuracy per 100 tokens). Debate was slightly more accurate than Consensus, but with similar latency and token use.

TABLE 3. Aggregate results across both models and all prompts.

Mode	Accuracy (%)	Latency (s)	Avg Tokens	Efficiency
Solo (CM0)	64.9	3.5	222	29.2
Debate (CM1)	78.3	13.7	767	10.2
Consensus (CM2)	71.6	11.5	679	10.5

5.2. Quantitative Comparison by Model

We next divide by model (Table 4). Both GPT-4.1 and GPT-4o benefited from collaboration. GPT-4.1 showed consistently higher solo accuracy and slightly better consensus accuracy, while GPT-4o achieved lower latencies in collaborative modes. The relative ranking of modes is similar across models:

Debate > *Consensus* > *Solo* for accuracy.

Solo < *Consensus* < *Debate* for efficiency.

TABLE 4. Per-model, per-mode performance (both task categories).

Model & Mode	Accuracy (%)	Latency (s)	Avg Tokens	Efficiency
GPT-4.1				
Solo (CM0)	66.6	3.6	210	31.7
Debate (CM1)	80	13.9	746	10.7
Consensus (CM2)	73.3	11.8	635	11.5
GPT-4o				
Solo (CM0)	63.3	3.5	234	27
Debate (CM1)	76.6	13.6	788	9.7
Consensus (CM2)	70	11.3	691	10.1

5.3. Breakdown by Task Category

Table 5 separates Problem Solving from fact-checking. Consensus holds a clear improvement for Problem Solving. Debate shows its strength in fact-checking/Synthesis.

TABLE 5. Mode performance by task category (aggregated over both models).

Category & Mode	Accuracy (%)	Latency (s)	Avg Tokens	Efficiency
Problem Solving (15 prompts)				
Solo (CM0)	63.3	3.6	220	28.7
Debate (CM1)	76.6	13.7	765	10
Consensus (CM2)	76.6	12.0	650	11.7
fact-checking & Synthesis (15 prompts)				
Solo (CM0)	66.6	3.7	225	29.6
Debate (CM1)	83.3	13.8	770	10.8
Consensus (CM2)	70	11.2	675	10.3

5.4. Error Taxonomy

Manual inspection of outputs revealed recurring patterns:

(1) **Arithmetic slips and chain errors** (Solo > Debate > Consensus). Consensus reduced single-step slips via explicit checks.
(2) **Schema non-compliance.** In early runs, agents returned narrative instead of requested JSON keys (Consensus), causing aggregator ambiguity and occasional scoring penalties; adding stronger schema instructions mitigated this.
(3) **Verbosity bias in judging** (Debate). The judge sometimes preferred longer, confident answers over short but correct ones, increasing false positives on ambiguous prompts; same solution as earlier was adopted to mitigate this as well.

6. Discussion

6.1. Why Collaboration Helps

Two simple mechanisms explain the gains:

(1) **Role-play reduces cognitive load.** In *Consensus*, the *Researcher* writes a step-by-step plan while the *Verifier* checks the correctness of each step. Writing down steps and checks makes hidden leaps visible and spots possible arithmetic errors.
(2) **Debate exposes disagreements.** In *Debate*, the agents challenge each other on concrete claims (dates, definitions, limits). This makes contradictions show up that would be missed in *Solo*, improving the final result's veracity.

6.2. Why Collaboration Sometimes Fails

Collaboration is not always better:

- **The judge can prefer style over substance.** When both answers sound good, a longer but subtly wrong one may win (especially in *Debate*). Using a judgment based on points per criterion, not by overall impression, and asking for *final answers only* can reduce this bias.
- **Shared mistakes spread.** If both agents begin with the same flawed assumption, their outputs can reinforce the error and the aggregator has little basis for correction. We collect explicit *confidence* with each answer to surface uncertainty. Encouraging diversity in perspectives helps reduce the risk of shared mistakes.
- **Coordination costs time and tokens.** Collaborative modes require multiple turns, leading to approximately 3× higher latency and token usage. For short, factual tasks or when operating under tight budgets, *Solo* remains the more practical choice.

6.3. Practical Recommendations

(1) **Use Consensus for multi-step reasoning.** Adopt a two-role workflow: a *Researcher* produces a clear plan, and a *Verifier* checks each step for assumptions, calculations, and edge cases. This structured response improves accuracy on problems while adding only moderate latency and token cost.
(2) **Use Debate for checking facts across sources.** The adversarial exchange emphasises contradictions and weak assumptions before committing to a final synthesis. This is especially effective for factual synthesis and cross-source reconciliation.
(3) **Control verbosity.** Set strict limits on intermediate outputs (e.g., "$\leq$ 120 tokens" per turn) and require clear outputs that follow the specified schema (correct fields, types, and order). This keeps steps comparable across agents, and makes aggregation and scoring more reliable.

7. Conclusions

This study presents a systematic evaluation of collaborative reasoning in Large Language Models, explicitly comparing single-agent performance against homogeneous multi-agent frameworks. Our results confirm that assigning distinct roles, even to identical underlying models, significantly enhances robustness. Specifically, we observed an average **accuracy improvement of approximately 10%** across all tasks when moving from Solo to Collaborative settings.

Our analysis reveals that the type of collaboration matters. We found a distinct specialization in interaction modes: **Consensus** proved most effective for structured problem solving, where cooperative verification reduced arithmetic and logical slips. In contrast, **Debate** excelled in fact-checking and synthesis, where adversarial critique was necessary to surface hallucinations.

These performance gains, however, must be weighed against efficiency. Collaborative modes present a **3× to 4× increase in latency and token usage**. Therefore, we conclude that while "two minds" are indeed better than one for complex, high-stakes reasoning tasks, solo reasoning remains the optimal choice for simpler queries where speed and cost are prioritized.

Future work will focus on calibrating judges with explicit rubrics that define clear, granular scoring criteria, reducing the degree to which evaluation outcomes depend on the judge model's own stylistic preferences or implicit biases. Experimenting with held-out judges, models excluded from the collaborative loop and exposed only to the final output, would further help isolate the evaluation signal from familiarity effects that arise when the same model family is used for both generation and assessment. Together, these measures aim to better balance quality against verbosity, preventing the judge from rewarding longer answers that do not necessarily demonstrate deeper correctness. We also plan to explore heterogeneous agent pools by mixing agents that differ across providers, sampling temperatures, and prompting styles, making the pool less likely to converge on the same blind spots or systematic errors that a homogeneous group would share. The central challenge will be managing coherence as diversity increases: diffuse or mutually inconsistent responses risk undermining the very gains that heterogeneity is meant to provide. Developing strategies to harness diverse reasoning while keeping collaborative output focused and concise will therefore be a key priority.

Looking further ahead, two promising directions are:

(i) **Adaptive collaboration policies** that dynamically switch between Solo, Debate, and Consensus based on real-time confidence and budget signals. Rather than committing to a fixed collaboration mode for all tasks, an adaptive policy could route simpler or high-confidence queries directly to Solo execution, reserving the more expensive Debate or Consensus modes for tasks where the model's uncertainty exceeds a given threshold or where the stakes of an incorrect answer are higher. This would recover much of the efficiency lost to collaborative overhead while preserving the accuracy benefits where they matter most.

(ii) **Specialized roles with tool access** such as retrieval or computation agents, that can be integrated into the collaborative loop to further improve reliability. Currently, all roles operate purely on parametric knowledge, meaning factual errors arising from outdated or hallucinated information cannot be corrected within the interaction itself. Equipping a designated role with access to external tools, such as a search engine for fact retrieval or a code interpreter for precise numerical computation, would address this limitation directly. Such a setup would allow the collaborative pipeline to ground its reasoning in verified, up-to-date information rather than relying solely on what the underlying model has memorised, substantially increasing the trustworthiness of the final output on knowledge-intensive or quantitative tasks.

Acknowledgement

This scientific research is financially supported within the project 'System for Scanning and Mapping IP Resources in Romania for the Early Detection of Cyber Threats,' contract no. 25Sol(T25)/2024, funded under the PN IV Program, 5.6 – Challenges, Subprogram 5.6.3 – Solutions.

REFERENCES

[1] *Humza Naveed, Asad Ullah Khan, Shi Qiu, Muhammad Saqib, Saeed Anwar, Muhammad Usman, Naveed Akhtar, Nick Barnes, and Ajmal Mian.* A comprehensive overview of large language models. *ACM Transactions on Intelligent Systems and Technology*, **16**(5):1–72, 2025.

[2] *Tian Liang, Zhiwei He, Wenxiang Jiao, Xing Wang, Yan Wang, Rui Wang, Yujiu Yang, Shuming Shi, and Zhaopeng Tu.* Encouraging divergent thinking in large language models through multi-agent debate. *arXiv preprint arXiv:2305.19118*, 2023.

[3] *Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Shaokun Zhang, Erkang Zhu, Beibin Li, Li Jiang, Xiaoyun Zhang, and Chi Wang.* Autogen: Enabling next-gen llm applications via multi-agent conversation framework. *arXiv preprint arXiv:2308.08155*, **3**(4), 2023.

[4] *Zhihua Duan and Jialin Wang.* Enhancing multi-agent consensus through third-party llm integration: Analyzing uncertainty and mitigating hallucinations in large language models. In *2025 8th International Conference on Advanced Algorithms and Control Engineering (ICAACE)*, pages 2222–2227. IEEE, 2025.

[5] *Shangbin Feng, Weijia Shi, Yike Wang, Wenxuan Ding, Vidhisha Balachandran, and Yulia Tsvetkov.* Don't hallucinate, abstain: Identifying llm knowledge gaps via multi-llm collaboration. *arXiv preprint arXiv:2402.00367*, 2024.

[6] *Keqin Bao, Jizhi Zhang, Yang Zhang, Xinyue Huo, Chong Chen, and Fuli Feng.* Decoding matters: Addressing amplification bias and homogeneity issue for llm-based recommendation. *arXiv preprint arXiv:2406.14900*, 2024.

[7] *Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al.* Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, **35**:24824–24837, 2022.

[8] *Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou.* Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.

[9] *Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan.* Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, **36**:11809–11822, 2023.

[10] *Stephanie Lin, Jacob Hilton, and Owain Evans.* Truthfulqa: Measuring how models mimic human falsehoods. *arXiv preprint arXiv:2109.07958*, 2021.

[11] *Yilun Du, Shuang Li, Antonio Torralba, Joshua B Tenenbaum, and Igor Mordatch.* Improving factuality and reasoning in language models through multiagent debate, 2023. *URL https://arxiv. org/abs/2305.14325*, **3**, 2023.

[12] *Cosima Rughiniş, Ştefania Matei, and Andreas Corcaci.* Generative content analysis for policy research: Comparing llm reliability in analyzing institutional ai discourse. In *2025 25th International Conference on Control Systems and Computer Science (CSCS)*, pages 604–611. IEEE, 2025.

[13] *Cosima Rughiniş, Mihai Dascălu, and Susantha Rasnayake.* Genai reliability in content analysis: Assessing agreement between llms in measuring discursive violence. In

2025 25th International Conference on Control Systems and Computer Science (CSCS), pages 612–619. IEEE, 2025.

[14] *Ana Ioan, Daniel Rosner, and Alexandru Radovici.* Generative ai and inter-rater reliability: Llm consistency in coding orders of worth in digital political debates. In *2025 25th International Conference on Control Systems and Computer Science (CSCS)*, pages 633–640. IEEE, 2025.

[15] *Cosima Rughiniş, Simona-Nicoleta Vulpe, Dinu Ţurcanu, and Răzvan Rughiniş.* Ai at the knowledge gates: Institutional policies and hybrid configurations in universities and publishers. *Frontiers in Computer Science*, 7:1608276, 2025.

[16] *Alexandru-Andrei Ghiţă, Răzvan Rughiniş, and Dinu Ţurcanu.* Gamifying ai evaluation: How well do chatbots perform in cybersecurity challenges? In *2025 25th International Conference on Control Systems and Computer Science (CSCS)*, pages 620–625. IEEE, 2025.

[17] *D. G. Badea, S. D. Ciocirlan, R. V. Rughiniş, and D. Turcanu.* Asynchronous federated learning: Convergence and performance in heterogeneous environments. *U.P.B. Scientific Bulletin, Series C*, 86(4):19–30, 2024.

[18] *C. Sandescu, A. Dinisor, C. V. Vladescu, O. Grigorescu, D. Corlatescu, M. Dascalu, and R. Rughiniş.* Extracting exploits and attack vectors from cybersecurity news using nlp. *U.P.B. Scientific Bulletin, Series C*, 84(2):63–78, 2022.

[19] *C. Contasel, R. V. Rughiniş, D. C. Tranca, and D. Turcanu.* Enhancing e-health cybersecurity and resilience: Shifting from monolithic to microservices architecture. *U.P.B. Scientific Bulletin, Series C*, 87(1):21–34, 2025.

[20] *Stephanie Lin, Jacob Hilton, and Owain Evans.* Teaching models to express their uncertainty in words. *arXiv preprint arXiv:2205.14334*, 2022.

[21] *Mirac Suzgun, Teven Le Scao, Valentin Hofmann, Jason Wei, Yi Tay, Hyung Won Chung, Noam Shazeer, Denny Zhou, Quoc V. Le, and Barret Zoph.* Challenging large language models with grade-school math problems. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2022.

[22] *Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt.* Measuring mathematical problem solving with the math dataset. *NeurIPS*, 2021.

[23] *James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal.* Fever: a large-scale dataset for fact extraction and verification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 809–819, 2018.

[24] *David Wadden, Shanchuan Lin, Kyle Lo, Lucy Lu Wang, Madeleine van Zuylen, Arman Cohan, and Hannaneh Hajishirzi.* Fact or fiction: Verifying scientific claims. *arXiv preprint arXiv:2004.14974*, 2020.