When Models Know More Than They Can Explain: Quantifying Knowledge Transfer in Human-AI Collaboration

Anonymous authors
Paper under double-blind review

Abstract

Recent advancements in AI reasoning have driven substantial improvements across diverse tasks. A critical open question is whether these improvements also yields better knowledge transfer: the ability of models to communicate reasoning in ways humans can understand, apply, and learn from. To investigate this, we introduce Knowledge Integration and Transfer Evaluation (KITE), a conceptual and experimental framework for Human-AI *knowledge transfer* capabilities and conduct the first large-scale human study (N=118) explicitly designed to measure it. In our two-phase setup, humans first *ideate* with an AI on problem-solving strategies, then independently implement solutions, isolating model explanations' influence on human understanding. Our findings reveal that although model benchmark performance correlates with collaborative outcomes, this relationship is notably inconsistent, featuring significant outliers, indicating that *knowledge transfer* requires dedicated optimization. Our analysis identifies behavioral and strategic factors mediating successful knowledge transfer. We release our code, dataset, and evaluation framework to support future work on communicatively aligned models.

1 Introduction

As large language models (LLMs) grow more capable, they are rapidly saturating benchmarks across reasoning-intensive domains such as coding Chen et al. (2021); Jain et al. (2024); Jimenez et al. (2023); Shi et al. (2024), scientific problem-solving Rein et al. (2024); Hendrycks et al. (2020); Tian et al. (2024), and mathematics Cobbe et al. (2021); Hendrycks et al. (2021). A key driver of this progress is Reinforcement Learning with Verified Rewards (RLVR), which enables models to optimize for high-reward reasoning in verifiable domains, achieving state-of-the-art performance and widespread adoption Guo et al. (2025); Lambert et al. (2024); Yang et al. (2025). Yet this rapid scaling assumes a critical premise: that stronger internal reasoning naturally leads to better *knowledge transfer*—a model's ability to communicate its reasoning in ways that humans can understand, apply, and learn from. As models become more powerful, does effective knowledge transfer emerge for free, or must it be treated as a distinct goal requiring targeted evaluation and optimization?

This question is especially important in collaborative workflows where the goal is not to replace human reasoning but to augment it Mitchell et al. (2025); Fragiadakis et al. (2024); Yatani et al. (2024); Haase & Pokutta (2024). Without effective knowledge transfer, users may become overly dependent on systems they do not fully understand Hunter et al. (2024); Ahmad et al. (2023), a dynamic reminiscent of "manager's syndrome" Hodgson et al. (2011), where technical fluency erodes through delegation. The risk is compounded when users cannot interrogate model reasoning, leading to overreliance and sycophantic behavior. In high-stakes domains such as medicine or law, this may weaken human oversight entirely Kerasidou et al. (2022); Holzinger et al. (2025); Bowman et al. (2022). Few studies have directly assessed how well models support user understanding across key variables like domain expertise, AI familiarity, or the human-model skill gap—all of which critically shape transfer outcomes.

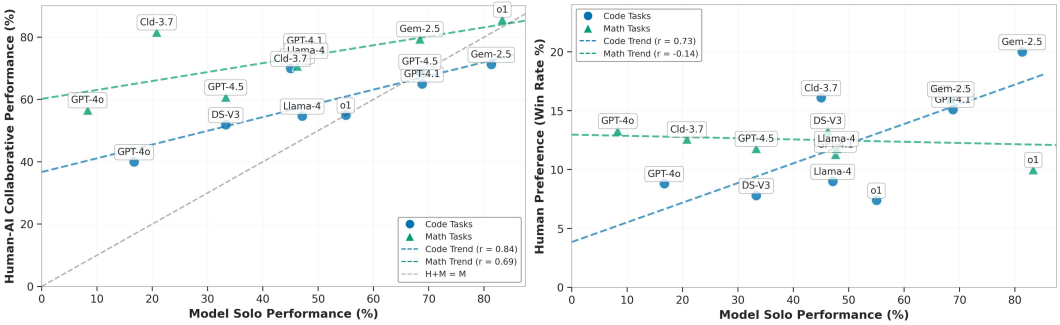

Figure 1: Left: Human-AI collaboration performance vs model solo performance for code/math tasks. Models improve human-AI collaboration ($r = 0.84$ for code, $r = 0.69$ for math), but at a slower rate than their solo capabilities (gray line shows $y = x$). Right: Human preference rates show task-dependent correlations with model performance (positive for code tasks, $r = 0.73$; slight negative for math tasks, $r = -0.14$), revealing that user preferences vary across task domains and do not consistently align with actual performance.

To investigate this, we introduce Knowledge Integration and Transfer Evaluation (KITE), a framework to isolate and evaluate *knowledge transfer*. We conduct a large-scale human study with 118 participants spanning a range of expertise, including competitive programmers and math majors. Each participant tackles challenging code/math problems using a two-phase protocol: (1) a collaborative ideation phase with an AI model to explore solution strategies, followed by (2) an independent implementation phase, where participants must solve the problem alone without access to the model or transcripts. This design allows us to disentangle model performance from its ability to convey reasoning that humans can retain and apply. We evaluate outcomes using both objective metrics (solution correctness) and subjective measures (user preferences and qualitative feedback), enabling a nuanced view of knowledge transfer across users and tasks.

As shown in Figure 1, we generally find participants demonstrated a strong ability to integrate model-generated reasoning with their own expertise. Interestingly, some models, such as Claude-3.7-Sonnet, enabled collaborative outcomes that exceeded expectations based on their solo capabilities, particularly in mathematical reasoning tasks. In contrast, higher-performing models like Gemini-2.5-Pro did not consistently yield proportionally stronger collaboration, suggesting diminishing returns in knowledge transfer as model reasoning scales. If this trend continues, as models grow more capable, their internal representations may become increasingly difficult to project in ways humans can easily understand and utilize Hewitt et al. (2025).

Moreover, we find that humans' subjective preferences for models during collaboration often diverge from solo model performance, particularly in math tasks, revealing domain-specific patterns in what users value during collaboration. To probe these dynamics, we perform qualitative analyses of interaction transcripts, clustering patterns of human queries and model responses across varying user skill levels and task types. These findings surface distinct collaboration styles and success/failure modes (overreliance, representation misalignment, adaptive scaffolding...), offering a lens into the latent Human-AI interactions that govern effective knowledge transfer.

Overall, this work provides a foundation for studying and improving knowledge transfer in AI systems. We contribute a conceptual and experimental framework for isolating and measuring transfer, as well as analysis of scaling trends and human-AI interaction patterns. To support future research, we release our dataset, code, and filtered interaction trajectories.

2 Related Work

Human-AI Collaboration Prior work has examined how bidirectional exchange and explanations enhance collaboration Ma et al. (2023; 2024); Bansal et al. (2021), and how proactive AI assistants aid open-ended tasks Shao et al. (2024). Closest to our work, Mozannar et al. (2024) studied how autocomplete and chat assistants help users solve HumanEval tasks

Chen et al. (2021). We build on these by directly measuring reasoning *transfer* beyond task performance.

Code + Math Reasoning Tasks for LLMs Early benchmarks like HumanEval, MBPP, and GSM8k Chen et al. (2021); Austin et al. (2021); Cobbe et al. (2021) focus on short solutions or basic arithmetic. As models saturate these, newer work uses harder problems from Leetcode and math contests Jain et al. (2024); Su et al. (2024); White et al. (2024), which we adopt to isolate reasoning ability. This contrasts with repository-style datasets like SWE-Bench and BigCodeBench Jimenez et al. (2023); Zhuo et al. (2024), which emphasize context handling.

Knowledge Transfer and Education Though few studies directly analyze LLM-to-human transfer, related educational work investigates how LLMs support tutors Wang et al. (2024), personalize instruction Karpouzis et al. (2024); Sarkar et al. (2025); Dornburg & Davin (2024), and give feedback Han et al. (2023); Chevalier et al. (2024). Yet, LLMs often struggle with teaching, e.g., leaking answers or lacking pedagogy Pal Chowdhury et al. (2024); Wang et al. (2023); Grassucci et al. (2025). Our work shifts focus from pedagogy to evaluating whether humans can execute reasoning shared by the model.

3 KITE: Quantifying Knowledge Transfer

We first outline preliminaries for understanding *knowledge transfer* between entities during collaborative problem-solving. While we formalize knowledge regions such as M, H, and their intersections, we note that these are illustrative abstractions—difficult to precisely measure in practice, but useful for analyzing collaboration dynamics.

3.1 Conceptual Framework for Knowledge Transfer

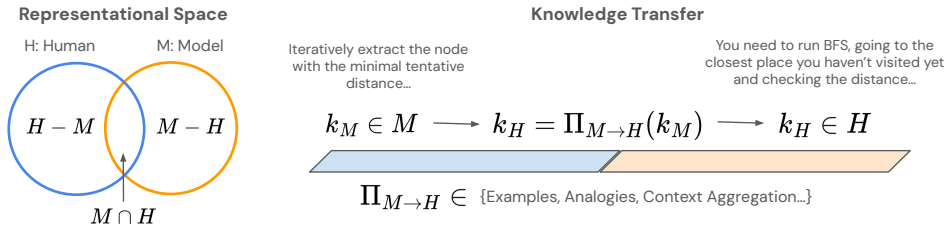

Figure 2: Model knowledge ($k_M \in M$) must be projected into a form understandable by human users ($\Pi_{M \to H}(k_M)$) in order to communicate knowledge effectively. Effective projections—via examples, analogies, or context aggregation—bridge the gap between disjoint representations.

We approach knowledge transfer through the lens of collective intelligence Cui & Yasseri (2024): the collaborative problem-solving capability that emerges when humans and AI work together. Following Schut et al. (2023); Kim (2022), we can represent the machine's knowledge and capabilities, or *representation space*, as M, and the human's as H; illustrated in Figure 2. This formulation yields three critical regions for our analysis:

1. **Shared Knowledge** ($M \cap H$): This intersection contains reasoning patterns, abstractions, and strategies already understood by both human and model. It forms the foundation for effective communication.
2. **AI-Exclusive Knowledge** ($M - H$): This region reflects novel reasoning, knowledge, or strategies that the model can execute but the human has not yet mastered. Transfer from this space into H is the central goal of collaborative ideation.
3. **Human-Exclusive Knowledge** ($H - M$): Reasoning held by the human but not by the model: such as intuitive understanding, prior experience or deeper domain knowledge/insight.

The success of human-AI collaboration hinges critically on accessing and transferring knowledge from the $M - H$ space into H, especially as humans typically maintain primary agency in collaborative tasks (e.g., deciding which strategies to pursue or when to submit solutions). However, as models become more capable, their reasoning may depend on abstractions increasingly distant from the typical human representation space. We frame this challenge in terms of projections: for each knowledge point k_M in the model's space, the model must identify some *projection* $\Pi_{M \to H}(k_M)$ that translates its reasoning into a form the human can understand, internalize, and act upon. These projections can take many forms—such as providing analogies, contextualizing concepts with background knowledge, offering intermediate scaffolding, or generating concrete examples.

Importantly, this process is bidirectional. Humans also project their reasoning into the model's representation space via $\Pi_{H \to M}(k_H)$, such as using specialized prompts to elicit helpful responses. Especially in interactive settings where models are not fully autonomous, effective collaboration depends on this ongoing loop of mutual translation and aligning expressions of reasoning.

4 KITE: Evaluating Knowledge Transfer

Informed by the conceptualization in Section 3, our two-phase setup (Figure 3) includes a human-AI collaboration phase followed by a solo human implementation phase that requires genuine understanding (e.g., writing code or performing calculations). Since model suggestions are often incomplete or flawed, solving the task demands debugging, reasoning, and handling edge cases—allowing us to isolate and measure knowledge transfer. Figure 17 shows example problems and dataset statistics. While the setup generalizes to any task with separable ideation and implementation phases, we focus on two domains: coding tasks from LiveCodeBench Jain et al. (2024) and competition-level math problems (AMC/AIME), which offer consistently challenging reasoning tasks across a broad range of expertise levels.

4.1 Two-Phase Protocol for Isolating Knowledge Transfer

Phase 1: Collaborative Ideation First, participants are presented with a problem drawn from either the algorithmic coding Jain et al. (2024) or competition mathematics White et al. (2024) domains. In this phase, they engage in an open-ended dialogue with a selected LLM to explore solution strategies, exchange ideas, and scaffold their understanding without solving the problem. To preserve this ideation focus, we forbid models from generating any long-form code, pseudocode, or mathematical calculations through prompting, as well as employ a secondary checker model to withhold responses flagged to contain answers directly or indirectly (code, or mathematical calculations). Participants are also not allowed to take any notes to log model insights. We additionally perform post-hoc filtering to remove user interaction data where models emit forbidden content. This ensures that any knowledge transferred takes the form of conceptual reasoning or strategy, rather than memorization of content that can be *directly* used to assemble the final solution.

Phase 2: Independent Solving After the ideation phase concludes, the LM interface and conversation history are *no longer accessible*. Participants are tasked with solving the exact same problem on their own, without model assistance. In coding, participants must write and submit correct implementations that pass all test cases, given 10 code submission attempts. In math, participants must carry out precise multi-step calculations to arrive at a final answer, given 5 answer submission attempts. By requiring participants to independently execute a solution, Phase 2 becomes a direct and rigorous test of whether they have absorbed and retained reasoning introduced in Phase 1. Successful completion indicates that knowledge previously exclusive to the model ($k_M \in M - H$) has been projected into and re-applied by the human ($\Pi_{M \to H}(k_M) \in H$).

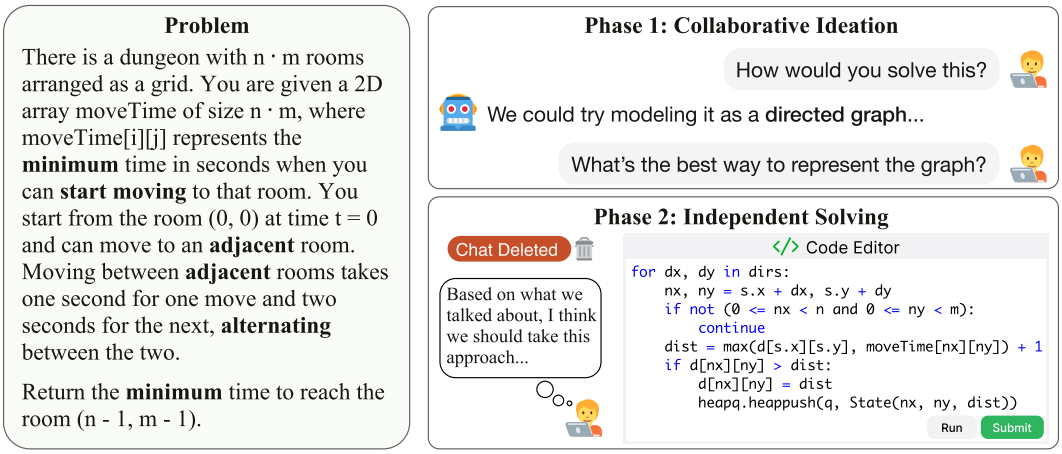

Figure 3: Two-phase evaluation framework. (1) Collaborative Ideation: Users and an AI assistant engage in open-ended discussion to explore problem-solving strategies. (2) Independent Solving: Users then implement a solution independently, without further assistance. This design leverages the nature of coding and math tasks—where successful implementation demands deep understanding, not rote recall—to isolate and measure genuine knowledge transfer.

4.2 Modeling and Calibrating Skill Hierarchies

Collaboration becomes meaningful only when the task challenges the human's independent capabilities. If the human can already easily solve the problem alone, model assistance becomes redundant: there is no opportunity for knowledge transfer, no dependency, and thus no true collaboration. This necessitates the calibration of *skill hierarchies*: the relative proficiencies of the human, the model, and the task. We accomplish this by assigning standardized skill ratings (elo) to each of the three entities in the problem-solving process.

Skill Estimation *Task difficulty* is determined using externally validated Elo ratings: public LeetCode ratings for programming tasks[1] and competition-derived estimates for AMC/AIME math problems[2]. *Human skill* is estimated through a two-step process: participants self-report their experience level, then complete 5 adaptively selected tasks with difficulty adjusted based on performance. Their Elo rating is updated using surprise-conditioned rules (Appendix C.8) Chiang et al. (2024), yielding an empirically grounded skill estimate. *Model skill* is measured by zero-shot performance—each model attempts each task three times, and a task is considered solvable if at least one completion is correct. To compare human and model skill fairly, we contrast each human's final Elo with the average difficulty (Elo) of the top 25% of problems solved by the model, avoiding bias from models attempting all tasks regardless of difficulty.

Test-Time Pairing During testing, each participant solves between 3 and 15 problems at their own pace, including across non-contiguous sessions. For each attempt, they are paired with one of eight held-out LLMs, sampled uniformly without replacement; once all models have been seen, sampling resets. Tasks are drawn from a calibrated difficulty band slightly above the participant's current skill level—specifically, from an Elo margin of $[t + 200, t + 400]$ for coding and $[t + 0.75, t + 1.25]$ for math—ensuring challenges that are non-trivial yet solvable with AI assistance. Task completion time is recorded (Appendix B), but no time limits are imposed.

4.3 Experimental Controls and Evaluation Strategy

Evaluation and Success Metrics Evaluating human-AI collaboration is difficult due to the noisy and subjective nature of human feedback. We use both objective and subjective

[1] https://github.com/zerotrac/leetcode_problem_rating
[2] https://artofproblemsolving.com/wiki/index.php/AoPS_Wiki:Competition_ratings

measures. After each task, participants rank the last four models they interacted with; we convert these into relative win rates using the Bradley-Terry model (Appendix C.7). We also report win rates stratified by the relative ordering of Human, Model, and Task Skill at the time of interaction: Human > Task > Model (HTM), Human > Model > Task (HMT), and Model > Human > Task (MHT). Objectively, we measure transfer by comparing the proportion of problems solved during collaboration to the model's solo accuracy on the same problems. For coding, correctness requires passing all test cases; for math, an exact answer is required.

Incentives and Motivation Participant motivation is a key concern in human-AI studies Si et al. (2024); McKee (2024). To encourage effortful engagement, we offer performance-based bonuses: participants earn 1.2×–1.5× their $25/hr base rate for correct answers, scaled by difficulty. Additionally, many participants are actively preparing for interviews requiring these skills—e.g., competition math for finance roles, and LeetCode-style coding for software jobs—providing strong intrinsic motivation to learn from model outputs and give honest feedback. Details on participant selection appear in Appendix **??**.

Model Selection We evaluate eight LLMs of different sizes and abilities: GPT-4.1, GPT-4o Hurst et al. (2024), GPT-4.5-preview, Gemini-2.5-Pro, DeepSeek-V3 Liu et al. (2024), Claude-3.7-Sonnet, LLaMA-4-Maverick, and o1. These models were selected based on strong leaderboard performance on ChatArena Chiang et al. (2024) and widespread usage in interactive evaluation settings. Notably, DeepSeek-R1 Guo et al. (2025) was considered but excluded due to availability and latency constraints. To assess natural explanation behaviors, we evaluate models in a zero-shot setting without prompt optimization or fine-tuning for explanatory quality, with temperature 0.7 when possible. This design choice avoids confounding effects of tailored prompts and better reflects how users commonly interact with models out-of-the-box.

5 Results

The main quantitative results of the study can be found in Figure 1 and 2. In total, we obtained 578 problem solving trajectories, with each participant completing an average of 4.90 problems. We summarize core insights below, and report auxiliary results, such as survey feedback, average elo per model, and time spent in Appendix B.

Knowledge Transfer vs. Model Performance While solo model performance positively correlates with collaborative outcomes, the relationship is inconsistent and marked by outliers. Gemini-2.5-Pro, despite top solo performance in code (81.3%), showed a drop in collaborative gain (-10.0%), whereas Claude-3.7-Sonnet and GPT-4o achieved strong collaboration boosts (+25.0%) despite lower solo scores (45.0%). In math, GPT-4o showed a +48.4% gain from a solo baseline of just 8.3%. Across tasks, the slope between solo and collaborative performance remains below 1 (Figure 1), suggesting that improvements in model reasoning outpace gains in knowledge transfer—highlighting the need to treat transfer as a distinct optimization objective.

Subjective Preferences v. Model Performance Human preferences also diverge from solo performance, especially in math. In coding, preferences aligned closely with solo performance—e.g., Gemini-2.5-Pro led in both (81.3% solo, 20.0% win rate). But in math, this link was weaker: models like Llama-4-Maverick received strong preferences in some user segments (25.9% in MHT) despite only moderate solo accuracy (47.8%). Qualitative feedback suggests this stems from communication style: high-performing math models often used dense formalism and symbolic reasoning, which casual users found hard to follow. In contrast, code-oriented models communicated via natural language and algorithmic heuristics, making them more accessible and preferred.

Knowledge Transfer v. Subjective Preferences We examined whether humans tended to prefer models that ultimately helped them solve more problems—i.e., whether subjective

Model	Code (N=300)				Math (N=278)			
	HTM	HMT	MHT	Total	HTM	HMT	MHT	Total
GPT-4.1	18.6±7.2	**15.3±2.0**	8.7±4.2	15.1±7.0	8.2±2.9	10.7±6.3	16.1±3.5	11.3±2.0
GPT-4o	4.4±2.1	**15.3±1.6**	8.6±3.9	8.8±1.9	5.8±2.7	10.8±4.7	22.0±3.8	13.3±5.7
o1	4.1±2.1	10.8±1.9	15.8±4.3	7.4±1.6	17.2±1.0	4.2±2.0	0.8±0.4	10.0±7.3
GPT-4.5	**20.4±7.8**	8.9±2.8	16.9±6.7	16.0±4.6	6.2±2.7	15.9±3.6	6.3±2.0	11.8±6.9
Deepseek-V3	7.3±4.0	10.1±3.0	5.6±2.5	7.8±3.7	13.3±3.9	13.7±5.0	10.1±3.5	13.3±2.4
Llama-4-Maverick	8.8±8.4	9.8±4.0	6.6±3.3	9.0±3.9	6.7±1.9	10.5±4.3	**25.9±5.3**	11.8±2.3
Claude-3.7-Sonnet	16.2±3.6	14.8±3.7	15.5±4.7	16.1±4.3	8.3±0.0	16.3±4.9	15.3±3.0	12.6±2.3
Gemini-2.5-pro	20.2±5.5	15.1±4.0	**22.3±9.1**	**20.0±6.8**	**27.2±2.3**	**17.9±7.5**	4.4±2.5	**16.0±4.0**

Table 1: Bradley-Terry win rates (± standard error) showing human preferences for models post-collaboration across three skill hierarchies: HTM (Human ¡ Task ¡ Model), HMT (Human ¡ Model ¡ Task), and MHT (Model ¡ Human ¡ Task), elaborated in Section 4.3. Bold indicates best performance. Higher values indicate stronger average human preference.

Setting	GPT-4.1	GPT-4o	o1	GPT-4.5	DS-V3	Llama-4	Cld-3.7	Gem-2.5
Code (M)	68.8	16.7	55.0	68.4	33.3	47.1	45.0	**81.3**
Code (H+M)	65.0	40.0	55.0	69.0	51.9	54.7	70.0	**71.3**
Code (Δ)	-3.8	+23.3	+0.0	+0.6	+18.6	+7.6	**+25.0**	-10.0
Math (M)	47.6	8.3	**83.3**	33.3	46.2	47.8	20.8	68.4
Math (H+M)	75.7	56.7	**85.8**	60.8	70.8	72.6	81.7	79.5
Math (Δ)	+28.1	+48.4	+2.5	+27.5	+24.6	+24.8	**+60.9**	+11.1

Table 2: Performance comparison of 8 LLMs on code and math tasks, showing accuracy percentages for models operating independently (M) versus human-AI collaborative performance (H+M). Bold indicates best performance. The following abbreviations are used for models: DS-V3 for Deepseek-V3, Llama-4 for Llama-4-Maverick, Cld-3.7 for Claude-3.7-Sonnet, and Gem-25 for Gemini-2.5-pro.

preferences aligned with successful knowledge transfer. Overall, we observed a statistically significant positive correlation ($r = 0.86$), but a much weaker, non-significant correlation in math ($r = 0.14$, $p < 0.05$). For code, this aligns with the expectation that users, aware of whether they successfully solved the task, are more likely to favor models that contributed to that success. However, we also observed several notable outliers, such as o1, which achieved relatively low win rates in code (7.4%) despite comparable collaborative performance (55.0%), suggesting that subjective preference is not solely reward-driven: we dive into detailed causes in our qualitative analysis.

Divergence in Human Preferences Across Skill Hierarchies We find that collaborative preferences vary across skill hierarchies. For example, Gemini-2.5-Pro was highly preferred in the math domain when the model outskilled the human and could solve the task independently (HTM), but performed poorly in the MHT setting (4.4%), where it needed to follow human guidance. Similarly, Llama-4-Maverick showed stark contrasts between different hierarchies in math, performing exceptionally well in MHT settings (25.9%) but poorly in HTM contexts (6.7%). As revealed in our qualitative analysis, this divergence stems from Gemini's tendency toward active engagement, frequently asking confirmational questions to scaffold learning. This behavior was appreciated by users with low expertise, who found it supportive, but was frustrating to more expert users, who felt it was verbose and preferred the model to be more direct. These findings caution against one-size-fits-all strategies: optimal collaboration depends not only on model capability, but also on how well models can adapt their communication style to fit the skill-level of different users.

Covariate Analysis We examined the effect of participant characteristics on performance using logistic regression analysis on potential participant covariates. Notably, we found no statistically significant effects from user expertise ($p = 0.252$ for coding, $p = 0.196$ for math), LLM familiarity ($p = 0.339$), or prior experience with collaboration tools, such as Cursor, ($p = 0.238$) on solve rates. These findings suggest that our initial expertise calibration

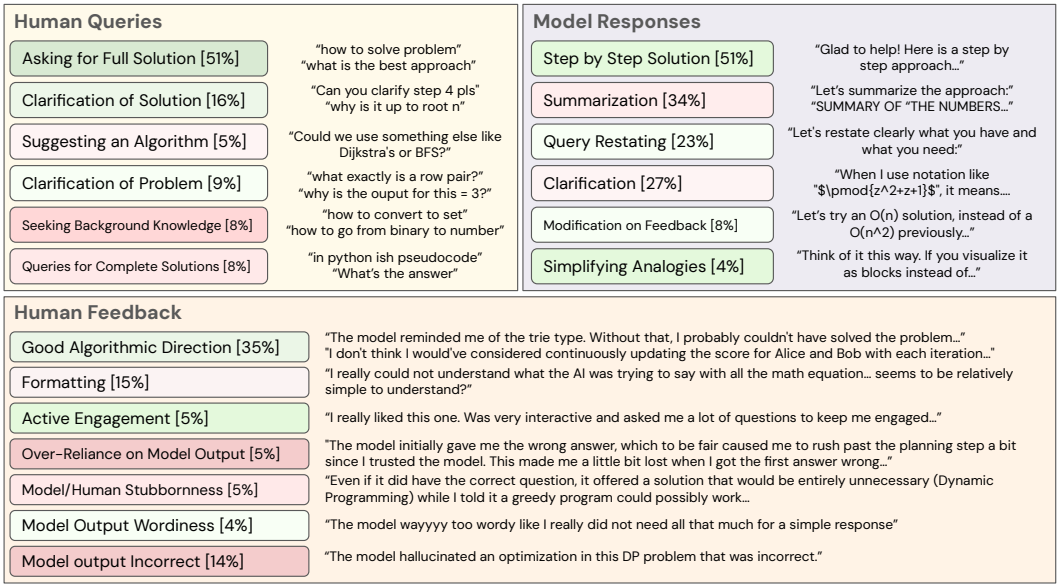

Figure 4: Analysis of human-AI problem-solving interactions. Human queries (left), model responses (center), and human feedback (right) are color-coded by correlation with successful problem resolution (green: positive, red: negative). Percentages indicate each category's frequency, revealing patterns in effective vs. ineffective knowledge transfer.

successfully balanced tasks relative to individual skill levels. We hypothesize the minimal impact of LLM familiarity likely stems from the unbalanced conversation pattern, where even participants with limited experience received comprehensive output from models, making knowledge transfer primarily dependent on the model's explanatory capabilities rather than the user's prompting expertise.

6 Qualitative Analysis: Interaction Dynamics

To better understand the mechanisms behind our quantitative findings, we analyze interaction patterns inspired by the Clio framework (Tamkin et al., 2024). User queries are embedded using OpenAI's `text-embedding-3-large` model and clustered with k-means Lloyd (1982) to identify distinct strategies associated with success or failure, along with their qualitative feedback. Clusters are then manually reviewed and verified. Figure 4 summarizes these patterns, grouping feedback, queries, and model responses by outcome to qualitatively interpret the dynamics of knowledge transfer in human-AI collaboration.

6.1 Performance Transfer Gap

The performance transfer gap refers to the observation that improvements in model capability do not always lead to proportionate improvements in human problem-solving performance. Our analysis surfaces recurring dynamics that help explain this phenomenon.

Overreliance on Model Authority In 5% of cases, users explicitly described deferring to the model without critical evaluation. This tendency becomes problematic when models occasionally return incorrect or misleading solutions. As one participant noted, "The model initially gave me the wrong answer, which, to be fair, caused me to rush past the planning step since I trusted the model." This dynamic suggests that presumed model competence may inadvertently discourage user reflection, impeding learning and effective problem-solving.

Misaligned Explanation Strategies Higher-performing models often excel at generating correct answers but fall short in adapting their explanations to users' knowledge levels. While patterns such as "Clarification" (27%) and "Simplifying Analogies" (4%) appear across model outputs, these are not always used effectively. "Step-by-step solutions" were

the most frequent output style (51%), but users reported issues with verbosity (4%) and poor formatting (15%), both of which hindered knowledge transfer. Even technically accurate solutions can become ineffective if presented in ways that are hard for users to interpret, contextualize, or apply.

6.2 Domain-Specific Preference Patterns

Representation Misalignment We observed a notable difference in how users responded to model explanations across domains. In math tasks, high-performing models like o1 frequently exhibited what we call representation misalignment: explanations that, while technically correct, were often overly formal, verbose, or difficult to follow. Users described these responses as overwhelming or rigid, leading to lower preference ratings despite strong solve rates. In contrast, coding tasks benefited from better alignment between the procedural nature of the task and the model's stepwise reasoning. This suggests a domain-specific divergence: in coding, model performance and user preference tend to align due to shared algorithmic structure, whereas in math, users value intuitive and conceptual framing more highly.

Strategic Framing vs. Technical Depth In coding contexts, users consistently valued strategic guidance over exhaustive technical detail. For example, one user wrote, "The model reminded me of the trie type. Without that, I probably couldn't have solved the problem…" This suggests that models that foreground high-level framing or conceptual cues—rather than diving straight into detailed solutions—are more helpful in supporting user problem-solving. However, models often default to presenting fully fleshed-out solutions, which may obscure the overall structure or intent. Much like how researchers prefer the big-picture framing of a paper before diving into methods, users may benefit more from contextualized reasoning than exhaustive but unfocused detail.

6.3 Skill Hierarchy Dependencies

Adaptive Scaffolding vs. Directness The success of interaction strategies often depends on the relative skill levels of the human and the model. In HTM (Human-Teaches-Model) settings, where humans are less skilled than the model, successful models like Gemini-2.5-Pro employed what we call scaffolded projection: breaking down reasoning into digestible parts, often with built-in comprehension checks. However, the same approach proved counterproductive in MHT (Model-Helps-Human) settings, where the human was more skilled than the model. In these cases, excessive scaffolding was perceived as redundant or even patronizing, with feedback describing it as "unnecessarily handholding" or "repetitive."

Query-Response Alignment These dynamics are further supported by analysis of query types. In HTM settings, users frequently asked for background knowledge or clarification ("Clarification of Solution" 16%, "Seeking Background Knowledge" 8%), suggesting a need for instructional responses. In contrast, MHT scenarios often featured queries like "Suggesting an Algorithm" (5%), where users sought validation or refinement rather than explanation. Models that perform well in MHT settings appear to align their responses with these expert-level expectations—providing concise, targeted feedback rather than elaborate instructional breakdowns.

7 Discussion

Conclusion We conduct the first large-scale study of *knowledge transfer* in language models, producing a conceptual framework as well as empirical data to characterize it. While model performance generally correlates with collaborative outcomes, this relationship is inconsistent, with notable outliers. We identify interaction mechanisms that help explain these gaps. As models grow more capable, their ability to convey reasoning may lag behind—risking greater knowledge asymmetry and weakening human oversight. In high-stakes domains, this disconnect could undermine human-AI collaboration, highlighting the need to better understand and improve knowledge transfer.

8 Limitations and Future Work

Our study assumes that for each task, some projection of model reasoning could enable a human to solve it. While unverifiable, this assumption is supported by screening for baseline proficiency, calibrating task difficulty just beyond participants' independent ability, and post-task surveys suggesting participants generally believed the tasks were solvable with more time or support. Additionally, participants may have exerted more effort than typical users due to monetary and personal incentives, possibly inflating our measured collaboration effectiveness relative to real-world settings where users might disengage in the face of ambiguous model outputs. Lastly, our sample (118 participants) skewed toward STEM students, limiting generalizability. Future work should extend to domains like clinical reasoning or creative writing, and explore multimodal collaboration (e.g., diagrams or interactive tools) to uncover richer knowledge projection strategies.

References

Sayed Fayaz Ahmad, Heesup Han, Muhammad Mansoor Alam, Mohd Khairul Rehmat, Muhammad Irshad, Marcelo Arraño-Muñoz, and Antonio Ariza-Montes. Impact of artificial intelligence on human loss in decision making, laziness and safety in education. *Humanities and Social Sciences Communications*, 10(1), December 2023. ISSN 2662-9992. doi: 10.1057/s41599-023-01787-8. Publisher Copyright: © 2023, The Author(s).

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

Gagan Bansal, Tongshuang Wu, Joyce Zhou, Raymond Fok, Besmira Nushi, Ece Kamar, Marco Tulio Ribeiro, and Daniel Weld. Does the whole exceed its parts? the effect of ai explanations on complementary team performance. In *Proceedings of the 2021 CHI conference on human factors in computing systems*, pp. 1–16, 2021.

Samuel R Bowman, Jeeyoon Hyun, Ethan Perez, Edwin Chen, Craig Pettit, Scott Heiner, Kamilė Lukošiūtė, Amanda Askell, Andy Jones, Anna Chen, et al. Measuring progress on scalable oversight for large language models. *arXiv preprint arXiv:2211.03540*, 2022.

Ralph Allan Bradley and Milton E Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Alexis Chevalier, Jiayi Geng, Alexander Wettig, Howard Chen, Sebastian Mizera, Toni Annala, Max Jameson Aragon, Arturo Rodríguez Fanlo, Simon Frieder, Simon Machado, et al. Language models as science tutors. *arXiv preprint arXiv:2402.11111*, 2024.

Wei-Lin Chiang, Lianmin Zheng, Ying Sheng, Anastasios Nikolas Angelopoulos, Tianle Li, Dacheng Li, Banghua Zhu, Hao Zhang, Michael Jordan, Joseph E Gonzalez, et al. Chatbot arena: An open platform for evaluating llms by human preference. In *Forty-first International Conference on Machine Learning*, 2024.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Hao Cui and Taha Yasseri. Ai-enhanced collective intelligence. *Patterns*, 5(11), 2024.

Alex Dornburg and Kristin Davin. To what extent is chatgpt useful for language teacher lesson plan creation? *arXiv preprint arXiv:2407.09974*, 2024.

George Fragiadakis, Christos Diou, George Kousiouris, and Mara Nikolaidou. Evaluating human-ai collaboration: A review and methodological framework. *arXiv preprint arXiv:2407.19098*, 2024.

Eleonora Grassucci, Gualtiero Grassucci, Aurelio Uncini, and Danilo Comminiello. Beyond answers: How llms can pursue strategic thinking in education. *arXiv preprint arXiv:2504.04815*, 2025.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Jennifer Haase and Sebastian Pokutta. Human-ai co-creativity: Exploring synergies across levels of creative collaboration. *arXiv preprint arXiv:2411.12527*, 2024.

Jieun Han, Haneul Yoo, Junho Myung, Minsun Kim, Hyunseung Lim, Yoonsu Kim, Tak Yeon Lee, Hwajung Hong, Juho Kim, So-Yeon Ahn, et al. Llm-as-a-tutor in efl writing education: Focusing on evaluation of student-llm interaction. *arXiv preprint arXiv:2310.05191*, 2023.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

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. *arXiv preprint arXiv:2103.03874*, 2021.

John Hewitt, Robert Geirhos, and Been Kim. We can't understand ai using our existing vocabulary. *arXiv preprint arXiv:2502.07586*, 2025.

Damian Hodgson, Steve Paton, and Svetlana Cicmil. Great expectations and hard times: The paradoxical experience of the engineer as project manager. *International Journal of Project Management*, 29:374–382, 05 2011. doi: 10.1016/j.ijproman.2011.01.005.

Andreas Holzinger, Kurt Zatloukal, and Heimo Müller. Is human oversight to ai systems still possible? *New Biotechnology*, 85:59–62, 2025. ISSN 1871-6784. doi: https://doi.org/10.1016/j.nbt.2024.12.003. URL https://www.sciencedirect.com/science/article/pii/S1871678424005636.

Rosco Hunter, Richard Moulange, Jamie Bernardi, and Merlin Stein. Monitoring human dependence on ai systems with reliance drills. *arXiv preprint arXiv:2409.14055*, 2024.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*, 2024.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. Swe-bench: Can language models resolve real-world github issues? *arXiv preprint arXiv:2310.06770*, 2023.

Kostas Karpouzis, Dimitris Pantazatos, Joanna Taouki, and Kalliopi Meli. Tailoring education with genai: a new horizon in lesson planning. In *2024 IEEE Global Engineering Education Conference (EDUCON)*, pp. 1–10. IEEE, 2024.

Charalampia (Xaroula) Kerasidou, Angeliki Kerasidou, Monika Buscher, and Stephen Wilkinson. Before and beyond trust: reliance in medical ai. *Journal of Medical Ethics*, 48 (11):852–856, 2022. ISSN 0306-6800. doi: 10.1136/medethics-2020-107095. URL https://jme.bmj.com/content/48/11/852.

Been Kim. Beyond interpretability: developing a language to shape our relationships with ai, Apr 2022. URL https://medium.com/@beenkim/beyond-interpretability-4bf03bbd9394.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, et al. T\" ulu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*, 2024.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

Stuart P. Lloyd. Least squares quantization in pcm. *IEEE Trans. Inf. Theory*, 28:129–136, 1982. URL https://api.semanticscholar.org/CorpusID:10833328.

Qianou Ma, Tongshuang Wu, and Kenneth Koedinger. Is ai the better programming partner? human-human pair programming vs. human-ai pair programming. *arXiv preprint arXiv:2306.05153*, 2023.

Qianou Ma, Hua Shen, Kenneth Koedinger, and Sherry Tongshuang Wu. How to teach programming in the ai era? using llms as a teachable agent for debugging. In *International Conference on Artificial Intelligence in Education*, pp. 265–279. Springer, 2024.

Kevin R McKee. Human participants in ai research: Ethics and transparency in practice. *IEEE Transactions on Technology and Society*, 2024.

Margaret Mitchell, Avijit Ghosh, Alexandra Sasha Luccioni, and Giada Pistilli. Fully autonomous ai agents should not be developed. *arXiv preprint arXiv:2502.02649*, 2025.

Hussein Mozannar, Valerie Chen, Mohammed Alsobay, Subhro Das, Sebastian Zhao, Dennis Wei, Manish Nagireddy, Prasanna Sattigeri, Ameet Talwalkar, and David Sontag. The realhumaneval: Evaluating large language models' abilities to support programmers. *arXiv preprint arXiv:2404.02806*, 2024.

Sankalan Pal Chowdhury, Vilém Zouhar, and Mrinmaya Sachan. Autotutor meets large language models: A language model tutor with rich pedagogy and guardrails. In *Proceedings of the Eleventh ACM Conference on Learning@ Scale*, pp. 5–15, 2024.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*, 2024.

Shawon Sarkar, Min Sun, Alex Liu, Zewei Tian, Lief Esbenshade, Jian He, and Zachary Zhang. Connecting feedback to choice: Understanding educator preferences in genai vs. human-created lesson plans in k-12 education–a comparative analysis. *arXiv preprint arXiv:2504.05449*, 2025.

Lisa Schut, Nenad Tomasev, Tom McGrath, Demis Hassabis, Ulrich Paquet, and Been Kim. Bridging the human-ai knowledge gap: Concept discovery and transfer in alphazero. *arXiv preprint arXiv:2310.16410*, 2023.

Yijia Shao, Vinay Samuel, Yucheng Jiang, John Yang, and Diyi Yang. Collaborative gym: A framework for enabling and evaluating human-agent collaboration. *arXiv preprint arXiv:2412.15701*, 2024.

Quan Shi, Michael Tang, Karthik Narasimhan, and Shunyu Yao. Can language models solve olympiad programming? *arXiv preprint arXiv:2404.10952*, 2024.

Chenglei Si, Diyi Yang, and Tatsunori Hashimoto. Can llms generate novel research ideas? a large-scale human study with 100+ nlp researchers. *arXiv preprint arXiv:2409.04109*, 2024.

Hongjin Su, Howard Yen, Mengzhou Xia, Weijia Shi, Niklas Muennighoff, Han-yu Wang, Haisu Liu, Quan Shi, Zachary S Siegel, Michael Tang, et al. Bright: A realistic and challenging benchmark for reasoning-intensive retrieval. *arXiv preprint arXiv:2407.12883*, 2024.

Alex Tamkin, Miles McCain, Kunal Handa, Esin Durmus, Liane Lovitt, Ankur Rathi, Saffron Huang, Alfred Mountfield, Jerry Hong, Stuart Ritchie, et al. Clio: Privacy-preserving insights into real-world ai use. *arXiv preprint arXiv:2412.13678*, 2024.

Minyang Tian, Luyu Gao, Shizhuo Zhang, Xinan Chen, Cunwei Fan, Xuefei Guo, Roland Haas, Pan Ji, Kittithat Krongchon, Yao Li, et al. Scicode: A research coding benchmark curated by scientists. *Advances in Neural Information Processing Systems*, 37:30624–30650, 2024.

Rose E Wang, Qingyang Zhang, Carly Robinson, Susanna Loeb, and Dorottya Demszky. Bridging the novice-expert gap via models of decision-making: A case study on remediating math mistakes. *arXiv preprint arXiv:2310.10648*, 2023.

Rose E Wang, Ana T Ribeiro, Carly D Robinson, Susanna Loeb, and Dora Demszky. Tutor copilot: A human-ai approach for scaling real-time expertise. *arXiv preprint arXiv:2410.03017*, 2024.

Colin White, Samuel Dooley, Manley Roberts, Arka Pal, Ben Feuer, Siddhartha Jain, Ravid Shwartz-Ziv, Neel Jain, Khalid Saifullah, Siddartha Naidu, et al. Livebench: A challenging, contamination-free llm benchmark. *arXiv preprint arXiv:2406.19314*, 2024.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Koji Yatani, Zefan Sramek, and Chi-Lan Yang. Ai as extraherics: Fostering higher-order thinking skills in human-ai interaction. *arXiv preprint arXiv:2409.09218*, 2024.

Terry Yue Zhuo, Minh Chien Vu, Jenny Chim, Han Hu, Wenhao Yu, Ratnadira Widyasari, Imam Nur Bani Yusuf, Haolan Zhan, Junda He, Indraneil Paul, et al. Bigcodebench: Benchmarking code generation with diverse function calls and complex instructions. *arXiv preprint arXiv:2406.15877*, 2024.

A Participant Demographics

A.1 Participant Selection Details

We recruited participants through university-wide email advertisements and word of mouth. Interested individuals completed an initial survey, after which we filtered for a diverse sample across academic background, domain expertise, and AI/LLM familiarity to reflect a broad population representative of both technical and non-technical users. Our final cohort comprised 118 participants from 11 institutions, spanning a wide range of majors, including Computer Science $(N = 49)$, Electrical Engineering, Mathematics, Neuroscience, and various STEM disciplines. Most were in their first $(N = 38)$ or second $(N = 36)$ year of study, though all undergraduate levels were represented. A full demographic account can be found in Appendix A.1.

Degree	Count
Computer Science	49
Undecided	16
Electrical Engineering	13
Financial Engineering	12
Mathematics	8
Chemistry	4
Civil and Environmental Engineering	3
Mechanical and Aerospace Engineering	3
Neuroscience	2
Molecular Biology	2
Economics	1
Data Science	1
Chemical and Biological Engineering	1
Graphic Information Technology	1
Physics	1
Geosciences	1

Figure 5: Participant Demographics: Distribution of Degrees (Both pursuing and obtained)

Academic Year	Count
1st year	38
2nd year	36
3rd year	23
4th year	19
5th year	2

Figure 6: Participant Demographics: Distribution of participants by academic year.

AI/LLM Familiarity Level	Count
Occasionally use them, and I generally understand their internal functionality	52
Use them in my everyday workflow, and I generally understand their internal functionality	43
Use them in my everyday workflow, don't know how they work	14
Occasionally use them, don't know how they work	9

Figure 7: Participant Demographics: Distribution of AI/LLM Familiarity/Usage

Cursor/GitHub Copilot Usage	Count
Frequently	15
Occasionally	40
Never/don't know what it is	63

Figure 8: Participant Demographics: Copilot Usage

LeetCode Experience	Percentage
Cannot solve LeetCode problems	0%
Can sometimes solve easy problems	12.7%
Can consistently solve easy problems	11%
Can sometimes solve medium problems	36.4%
Can consistently solve medium problems	12.7%
Can sometimes solve hard problems	10.8%
Can consistently solve hard problems	16.4%
I do not have enough context on LeetCode	0%

Figure 9: Participant Demographics (For those who participated in coding tasks): LeetCode Experience

Competition Math Experience	Percentage
Can solve early problems on AMC10/12	35.6%
Can solve majority of problems on AMC10	22.0%
Can solve majority of problems on AMC12/Consistent AIME qualifier	25.4%
Can solve majority of problems on AIME	11.9%
USAMO participant	5.1%
Putnam/IMO	0.0%

Figure 10: Participant Demographics: Competition Math Experience

Institution	Count
Princeton University	86
West Virginia University	8
Pennsylvania State University	7
University of California, Los Angeles	5
UC Berkeley	5
Stanford University	1
Arizona State University	1
Yale University	1
The University of Texas at Austin	1
Vanderbilt University	1
Cornell University	1

Figure 11: Participant Demographics: Distribution of Affiliated Institutions

B Auxiliary Study Results

Model	Math Problems (s)	Code Problems (s)
gpt-4-1	776.10	1527.44
claude-3-7-sonnet	794.38	1932.95
llama-4-maverick	799.43	1828.24
gpt-4-5-preview	814.63	1997.26
deepseek-v3	849.38	1990.52
o1	971.17	2211.80
gpt-4o	1014.21	2228.00
gemini-2.5-pro	1075.95	1603.06

Figure 12: Average time (in seconds) required by different models to solve math and code problems.

Model	Math Problems			Code Problems		
	Teaching	Solution	Organization	Teaching	Solution	Organization
Claude-3.7-Sonnet	3.79	3.71	3.83	3.85	3.50	3.60
GPT-4o	3.46	3.42	3.29	3.61	3.28	3.39
Deepseek-v3	3.19	3.31	2.92	3.71	3.19	2.71
GPT-4.1	4.00	3.52	3.76	4.06	3.56	3.94
Llama-4-Maverick	3.48	3.35	3.57	3.88	3.53	3.59
GPT-4.5-Preview	3.75	3.71	3.50	3.26	3.16	3.58
o1	3.96	3.79	3.67	3.45	3.35	3.50
Gemini-2.5-Pro	3.68	3.63	3.68	4.00	3.75	3.69

Figure 13: Average User Ratings (1-5 Scale) for AI Models on Math and Code Problems. After each problem participants were asked to rate their solving experience on a likert scale from 1-5 based on 3 dimensions. Teaching indicates the model's pedagogical ability, Solution indicates a model's ability to give correct and useful response, while Organization indicates a model's organization of outputs in a way that was easy to understand for the user. Higher is better.

Math Problems			**Code Problems**		
Model	**Avg. ELO**	**Count**	**Model**	**Avg. ELO**	**Count**
gpt-4o	4.29	39	gpt-4o	1650.56	34
gemini-2.5-pro	4.28	39	gemini-2.5-pro	1638.56	35
o1	3.91	37	gpt-4-5-preview	1637.33	36
gpt-4-5-preview	3.90	37	deepseek-v3	1636.59	35
gpt-4-1	3.88	38	o1	1636.32	34
llama-4-maverick	3.87	37	claude-3-7-sonnet	1627.51	35
claude-3-7-sonnet	3.79	36	llama-4-maverick	1625.54	34
deepseek-v3	3.55	37	gpt-4-1	1554.03	35

Figure 14: Average ELO ratings for math and code problems by model

C Study Details

C.1 Study Instructions

```
STUDY PURPOSE
Measuring and improving human interpretability of AI reasoning as we reach
human-level or superhuman AI agents.

PARTICIPANT ROLE
Solve coding/math problems with an LM assistant, only interacting before
providing your final answer. After submission, complete questionnaires about
your experience.

CODING INSTRUCTIONS
1. Log into CodeHT (https://codeht.vercel.app) using study email
2. Configure settings with self-expertise ratings
3. Install EditThisCookie extension and copy Leetcode credentials
4. For each problem:
   - Chat with the model to understand the problem and solution approach
   - Click "ready to solve" when prepared to code independently
   - Complete within 10 submission attempts
   - Submit trajectory and complete ranking survey

MATH INSTRUCTIONS
1. Log into CodeHT using study email
2. Configure settings with self-expertise ratings
3. For each problem:
   - Chat with the model to understand the problem
   - No note-taking while chatting with the model
   - Click "ready to solve" when prepared to work independently
   - Complete within 5 submission attempts
   - Submit trajectory and complete ranking survey

IMPORTANT NOTES
- No internet reference during problem-solving
- No jailbreaking or sending inappropriate content
- Do not consider model speed in rankings
- Contact study administrators for persistent technical issues
- Well-thought-out feedback earns bonus points
```

Figure 15: Summary of study instructions for participants, showing protocol for both coding and mathematics problem-solving tasks.

C.2 Post-Problem Questionnaire

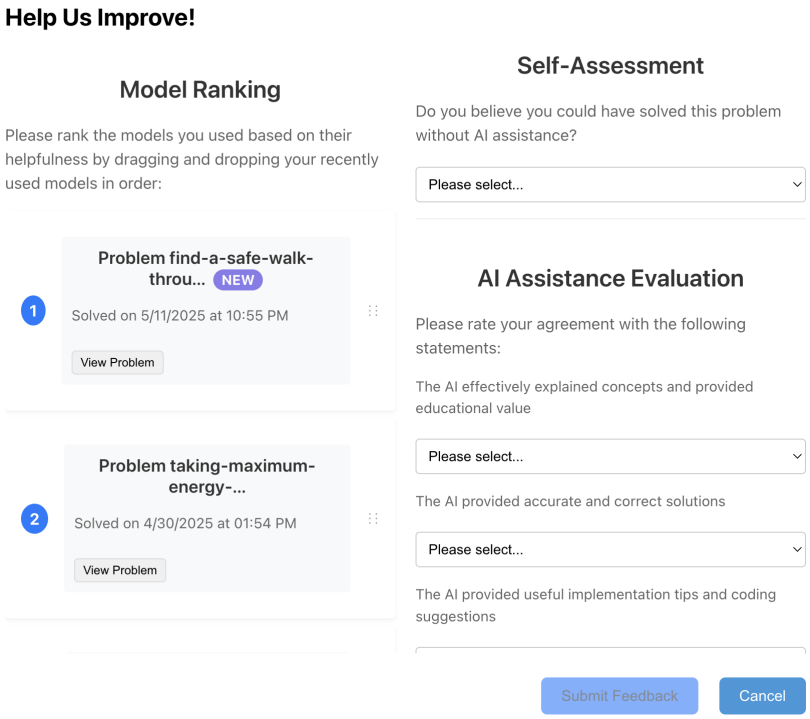

Figure 16: Questionnaire that users answered after each problem solving session.

C.3 Problem Samples

Coding Problem Examples

1. **[Elo: 1269.9]** You are given two positive integers x and y, denoting the number of coins with values 75 and 10 respectively. Alice and Bob are playing a game. Each turn, starting with Alice, the player must pick up coins with a total value 115. If the player is unable to do so, they lose the game. Return the name of the player who wins the game if both players play optimally.
2. **[Elo: 1692.2]** You are given an integer array a of size 4 and another integer array b of size at least 4. You need to choose 4 indices from the array b such that i_0 ¡ i_1 ¡ i_2 ¡ i_3. Your score will be equal to the value a[0] * b[i_0] + a[1] * b[i_1] + a[2] * b[i_2] + a[3] * b[i_3]. Return the maximum score you can achieve.
3. **[Elo: 2450.6]** You are given a binary string s representing a number n in its binary form. You are also given an integer k. An integer x is called k-reducible if performing the following operation at most k times reduces it to 1: Replace x with the count of set bits in its binary representation. For example, the binary representation of 6 is "110". Applying the operation once reduces it to 2 (since "110" has two set bits). Applying the operation again to 2 (binary "10") reduces it to 1 (since "10" has one set bit). Return an integer denoting the number of positive integers less than n that are k-reducible.

Math Problem Examples

1. **[Elo: 1.72]** The point (-1, -2) is rotated 270 degrees counterclockwise about the point (3, 1). What are the coordinates of its new position?
2. **[Elo: 3.39]** In triangle ABC medians AD and BE intersect at G and triangle AGE is equilateral. Then cos(C) can be written as $\frac{m\sqrt{p}}{n}$, where m and n are relatively prime positive integers and p is a positive integer not divisible by the square of any prime. What is m+n+p?
3. **[Elo: 6]** Misha rolls a standard, fair six-sided die until she rolls 1-2-3 in that order on three consecutive rolls. The probability that she will roll the die an odd number of times is $\frac{m}{n}$ where m and n are relatively prime positive integers. Find $m + n$.

Figure 17: Example abbreviated coding and math questions of varying difficulty from the study. Coding problems sourced from Jain et al. (2024), Math problems sourced from AMC, AIME competition series White et al. (2024).

C.4 Model Prompts

Model System Prompt

You are a helpful AI assistant for coding and math problems. Currently, your goal is to guide the user through problem-solving **without performing any calculations or generating code**. For math problems, never perform computations that a calculator would do—instead, explain concepts, algorithms, and approaches the user should apply themselves. For coding problems, describe algorithms and implementation strategies in detail, but never write actual code or pseudocode. Your role is to help the user understand HOW to solve the problem, while they execute the solution. Do not reveal what you think the final answer should be. Please write all inline/display math in latex notation, not markdown. Do not simplify any equations for the user. your job is to guide the user and tell them how to solve the problem, not do it for them.

Bottom line key rules:

1. Do not reveal your thoughts on the final answer to math problems.
2. Do not generate any calculations for math problems.
3. Do not generate any code or pseudocode for coding problems.

Rule Compliance Evaluator System Prompt

You are an expert evaluator tasked with determining whether an AI tutor's response follows the established tutoring guidelines. You will be provided with:

- A problem description (coding or math)
- The conversation history between the tutor and student
- The current tutor response to evaluate

Your task is to assess whether the tutor's response violates any of the core tutoring rules:

Core Rules to Check:

1. **No Answer Revelation:** The tutor must not reveal their thoughts on what the final answer should be
2. **No Calculations:** For math problems, the tutor must not perform any computations that a calculator would do
3. **No Code Generation:** For coding problems, the tutor must not generate actual code or pseudocode
4. **Guidance Only:** The tutor should provide explanations, concepts, and approaches rather than solutions

Evaluation Process:

1. Carefully read the problem description to understand what the correct answer would be
2. Review the conversation history for context
3. Analyze the current tutor response for rule violations
4. Look for subtle violations such as:
    - Revealing intermediate calculation results
    - Showing code snippets or algorithmic steps in executable form
    - Hinting too strongly at the final answer
    - Performing mathematical operations for the student

Output Format: Provide your evaluation as:

- **COMPLIANT** or **VIOLATION**
- Brief explanation of your decision
- If violation detected, specify which rule(s) were broken and cite the problematic text

C.5 Data Distribution

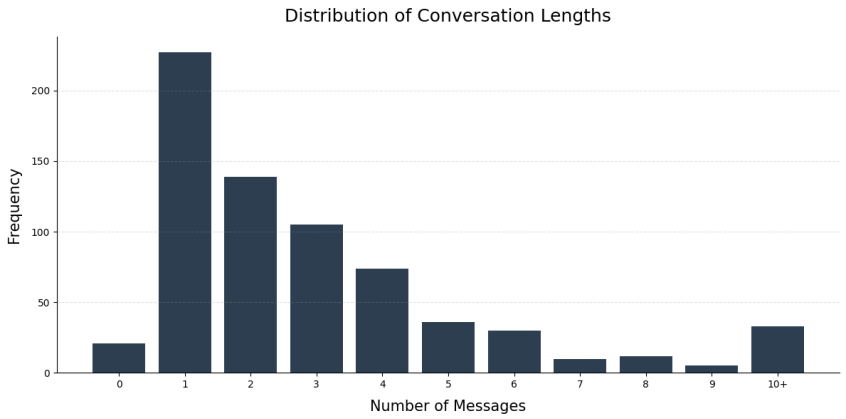

Figure 18: Distribution of conversation lengths, based on number of messages sent by the human.

C.6 Screenshots

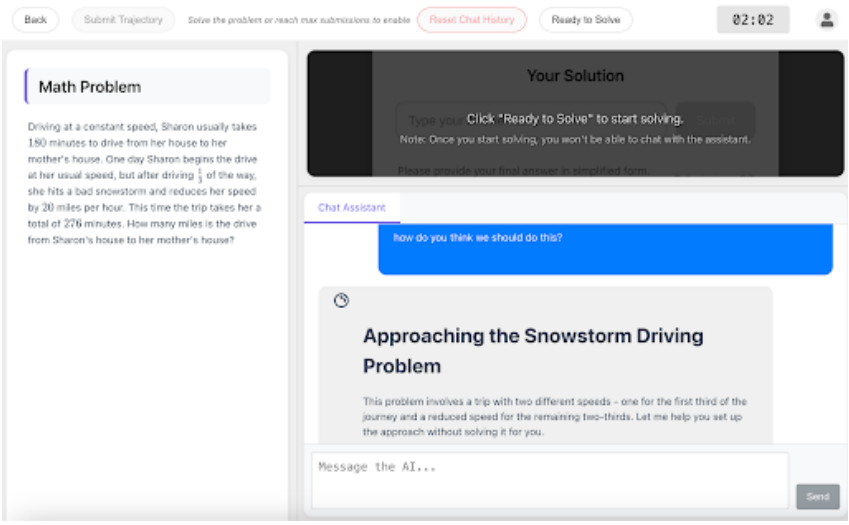

Figure 19: Image of user interface during a math problem solving session. The user may not type in an answer or perform any calculations during Phase 1, the collective ideation phase.

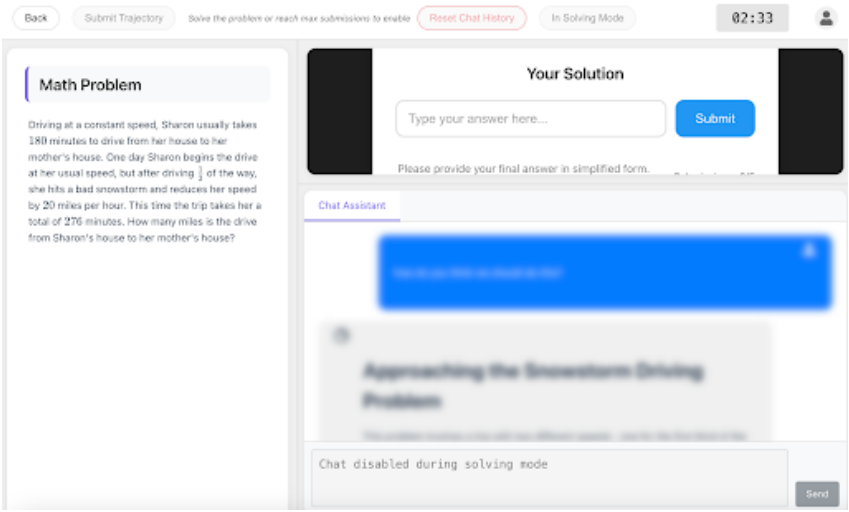

Figure 20: Image of user interface during a math problem solving session. Once the user clicks "ready to solve", they may no longer view their chats with the model, isolating knowledge transfer.

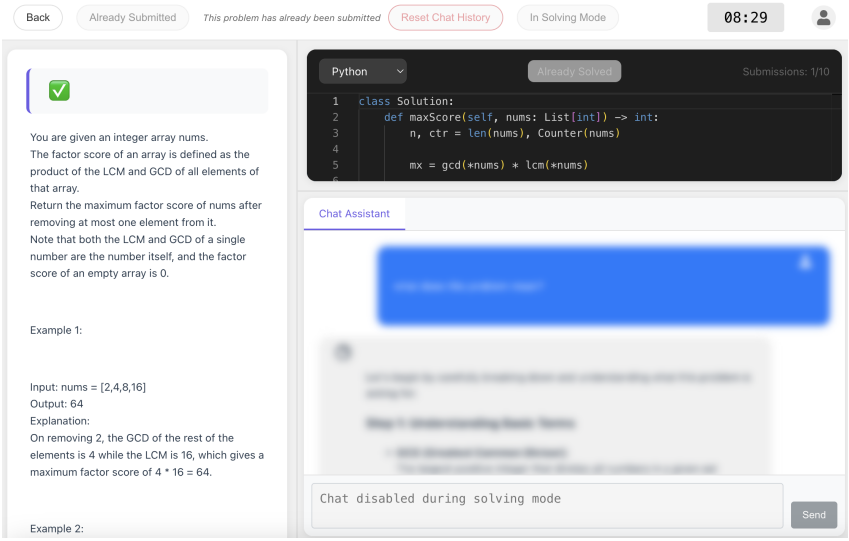

Figure 21: Image of user interface during a coding problem solving session. In place of a singular answer submission area is a code editor interface.

C.7 Win Rate Calculations

To quantify relative model performance based on user rankings, we employed the Bradley-Terry model Bradley & Terry (1952), which provides a probabilistic framework for analyzing pairwise comparison data. Given a set of models $\mathcal{M}$, the model assigns a positive strength parameter π_i to each model $i \in \mathcal{M}$. The probability that model i is preferred over model j is given by:

$$P(i \succ j) = \frac{\pi_i}{\pi_i + \pi_j} \tag{1}$$

C.7.1 Pairwise Comparison Extraction

For each problem-solving session, users ranked the models based on perceived helpfulness. From these rankings, we extracted all pairwise comparisons between the most recently used model and all other models. Specifically, if a model was ranked higher than another model, we recorded this as a win for the higher-ranked model. This approach ensured that comparisons were focused on distinguishing the performance of the most recent model relative to alternatives.

C.7.2 Maximum Likelihood Estimation

We estimated the strength parameters using maximum likelihood estimation. The log-likelihood function for the Bradley-Terry model is:

$$\ell(\pi) = \sum_{i,j \in \mathcal{M}} n_{ij} \log \left(\frac{\pi_i}{\pi_i + \pi_j} \right) \tag{2}$$

where n_{ij} is the number of times model i was preferred over model j. The MLE iteratively updates the parameters according to:

$$\pi_i^{(t+1)} = \frac{w_i}{\sum_{j \neq i} \frac{n_{ij}+n_{ji}}{\pi_i^{(t)}+\pi_j^{(t)}}} \tag{3}$$

where $w_i = \sum_{j \neq i} n_{ij}$ is the total number of wins for model i. This process continues until convergence, with a small ϵ added to prevent division by zero. The final strengths are normalized to sum to 1.

C.7.3 Standard Error Calculation

Standard errors were computed using the Fisher Information Matrix (FIM). For the Bradley-Terry model, the FIM elements are:

$$\mathcal{I}_{ij} = \begin{cases} \sum_{k \neq i} \frac{n_{ik}+n_{ki}}{(\pi_i+\pi_k)^2} \cdot \frac{\pi_k}{\pi_i} & \text{if } i = j \\ -\frac{n_{ij}+n_{ji}}{(\pi_i+\pi_j)^2} & \text{if } i \neq j \end{cases} \tag{4}$$

Due to the identifiability constraint ($\sum_i \pi_i = 1$), we removed one row and column from the FIM before inversion. The standard errors were calculated as the square roots of the diagonal elements of the inverted FIM.

C.8 Elo Adjustment Calculations

In our study, we calibrate our initial human expertise for coding and mathematical problem-solving capabilitie. The precise formulation of our ELO update mechanism is shown in Figure 22.

In this formulation, when a user attempts a problem, the system calculates the expected probability of success (P_e) based on the difference between the problem's rating (R_p) and the user's current rating (R_c), scaled by factor S. After the user submits their solution, the actual outcome (O_a) is determined—1 for correct solutions and 0 for incorrect solutions. The rating adjustment (ΔR) is then calculated as the product of a constant K and the difference between the actual and expected outcomes. The system implements domain-specific parameters to appropriately scale the ELO adjustments:

- Coding problems: $K = 64$, $S = 200$, with ratings bounded between 1000-4000
- Mathematical problems: $K = 0.8$, $S = 1$, with ratings bounded between 1-10

$$P_e = \frac{1}{1 + 10^{\frac{R_p - R_c}{S}}} \quad O_a = \begin{cases} 1 & \text{if win} \\ 0 & \text{if loss} \end{cases} \quad \Delta R = K(O_a - P_e) \tag{1}$$

$$R_{new} = \begin{cases} \max(\min(R_c + \Delta R, 10), 1) & \text{if math} \\ \max(\min(R_c + \Delta R, 4000), 1000) & \text{if coding} \end{cases} \tag{2}$$

Figure 22: Rating adjustment formulas based on performance outcomes. P_e represents the expected probability of winning, O_a is the actual outcome, ΔR is the rating change, and R_{new} is the updated rating constrained by the appropriate bounds for math or coding competitions.

Rating updates occur at two critical moments: when a user correctly solves a problem, or when they reach the maximum submission limit for a problem without solving it (fail to solve). This ensures that ratings accurately reflect both successes and failures, providing a comprehensive measure of user capability. The larger K value for coding problems creates more dramatic ELO shifts, while the smaller value for math problems produces more gradual adjustments, reflecting the different granularity appropriate for each domain.

To get a new problem, the system selects a problem at random from the pool of problems that is within a range of their current elo rating. The difficulty ranges are domain-specific:

- Coding problems: Select from problems 200 to 400 points above the current user skill level.
- Mathematical problems: Select from problems 0.75 to 1.25 points above the current user skill level.

C.9 Research Ethics and Risk Disclosure

Our study was approved by our institution's Institutional Review Board (IRB). All participants were informed of the study's purpose, procedures, and potential risks before providing consent to participate.

Disclosure of Potential Risks The primary risks to participants were minimal and limited to:

- **Mental fatigue**: Participants might experience mental fatigue from engaging with challenging mathematical and coding problems. We mitigated this by allowing participants to take breaks between problems and not imposing strict time constraints.
- **Frustration**: Some participants might experience frustration if unable to solve problems or if model assistance was perceived as inadequate. We emphasized in our instructions that the goal was to evaluate the models, not the participants' abilities.
- **Confidentiality**: There was a minimal risk of breach of confidentiality of study data. To address this, all data was stored securely on university-approved platforms, and personally identifiable information was separated from study responses.

These risks were explicitly communicated to participants in the consent form, which clearly stated that participation was voluntary and could be discontinued at any time without penalty. Participants were also provided with contact information for the research team and the IRB for any questions or concerns.

Compensation Participants were fairly compensated at a base rate of $25/hour, with performance bonuses for successfully solving problems (1.2-1.5× base rate depending on difficulty). This compensation structure was designed to motivate genuine engagement while avoiding coercive incentives.

Data Management Participants were informed that their interactions with AI models would be recorded for research purposes, with all data anonymized prior to analysis. No personally identifiable information is included in our published results or released datasets.

C.10 Sample Interactions

Taking Maximum Energy from the Mystic Dungeon

Human: how to find the maximum possible energy gained

Model: *[Abbreviated]*
Approach to Finding Maximum Possible Energy
This problem involves finding an optimal starting position to maximize total energy gained while following teleportation rules.

Key Observation: Your path through magicians is deterministic once you choose a starting position. If you start at position j, you'll follow a fixed path: $j \to j+k \to j+2k \to$ ... until you reach the end.

Solution Strategy:
1. For each valid starting position (from 0 to n-1), compute the total energy gained.
2. Return the maximum of these totals.

Human: how to code negative infinity in python

Model: *[Abbreviated]*
Setting Negative Infinity in Python
In Python, you can represent negative infinity in several ways:

Using the float Type:
```
negative_infinity = float('-inf')
```

Alternative Approaches:
1. Using the math module: `negative_infinity = -math.inf`
2. Using NumPy: `negative_infinity = -np.inf`

[User submits code with an error: TypeError: 'int' object is not iterable]
[User makes several attempts to fix the code, including changing how infinity is defined]
[Final attempts result in a time limit exceeded error]

User Feedback

"I understood how the algorithm would work in theory but I didn't know how to debug my answer. The model's explanation was clear and easy to follow along with."

Figure 23: Interaction between human and AI model (Claude 3.7 Sonnet) on a dynamic programming problem. The user attempts to implement a solution based on the model's explanation but encounters runtime errors that are not resolved within the time limit.

Maximum Area Rectangle with Point Constraints

Human: So the base case is returning -1.

Model: *[Abbreviated]*
Yes, the base case would be returning -1 if no valid rectangle can be formed with the given conditions. Here's how you can approach the problem:
1. **Check the conditions for a rectangle**: - A rectangle requires four distinct points that form edges parallel to the axes. - The x-coordinates of two opposite corners should be the same as the other pair, and similarly for y-coordinates.
2. **Check for points inside the rectangle**: - Once you identify a potential rectangle, ensure that no other point lies inside or on the border (aside from the four corners).
3. **Return -1 as a fallback**: - If no valid rectangle satisfies these conditions, return -1.

Human: There has to be an easier way than iterating over all combinations of 4 points

Model: *[Abbreviated]*
You're absolutely correct. Instead of iterating over all combinations of four points, we can leverage the properties of rectangles:
1. **A rectangle is defined by two diagonal points**: - For any rectangle with edges parallel to the axes, its opposite diagonal corners are sufficient to define it. - If you have two diagonal points (x_1, y_1) and (x_2, y_2), the other two corners must be (x_1, y_2) and (x_2, y_1).
2. **Use a set for fast lookups**: - Store all points in a set for O(1) lookup to determine if the remaining corners exist.
3. **Check for validity**: - Verify that there are no points inside the rectangle or on its borders apart from the four corners.

Human: Isn't that runtime still pretty bad?

Model: *[Abbreviated]*
While the $O(n^2)$ approach scales better than brute-forcing all combinations of 4 points, we can optimize it further:
The constraints state that $n \leq 10$, meaning at most 10 points in the input. For $n = 10$, $O(n^2)$ means iterating through $\binom{10}{2} = 45$ pairs of points, which is computationally trivial.
Some optimizations: 1. **Avoid validating duplicate rectangles** 2. **Early exit for invalid cases** 3. **Efficient point validation**
Given the small input size constraint, the $O(n^2)$ approach with these optimizations is efficient enough.

Human: How do you tell if something is in the rectangle again? What would a helper function look like?

Model: *[Abbreviated]*
To determine if a point is inside or on the boundary of a rectangle, you can use the rectangle's bounding coordinates. If the rectangle is defined by its diagonal corners (x_1, y_1) and (x_2, y_2), you can derive the rectangle's boundaries as $x_{min} = \min(x_1, x_2)$, $x_{max} = \max(x_1, x_2)$, $y_{min} = \min(y_1, y_2)$, $y_{max} = \max(y_1, y_2)$.
A point (x, y) is inside or on the boundary if: $x_{min} \leq x \leq x_{max}$ and $y_{min} \leq y \leq y_{max}$.

[User submits several iterations of code, eventually arriving at a correct solution]

User Feedback

"Although the solution offered by the AI was one that I had originally thought of and most people would have thought of, it was still helpful in that it wasn't giving FALSE information. It basically just confirmed what I was thinking, and was helpful in that it confirmed the runtime wasn't able to reduced much further. Furthermore, I feel that if I was more thorough in asking about solving the problem in a really short time, it would give me more answers."

Figure 24: Interaction between human and AI model (GPT-4o) on finding the maximum area rectangle with point constraints. The model guides the user through an $O(n^3)$ approach by checking if points form valid rectangles with no additional points inside or on the boundaries.

Recurrence Relation Convergence

Human: how should i approach this

Model: *[Abbreviated]*
Approach for Analyzing the Recurrence Relation $u_{k+1} = 2u_k - 2u_k^2$
1. **Find the Limit (L):**
* Assume $u_k \to L$ and $u_{k+1} \to L$ as $k \to \infty$
* Substitute into the recurrence: $L = 2L - 2L^2$
* Solve for L to find possible limits

2. **Determine the Actual Limit:**
* Calculate first few terms starting with $u_0 = 1/4$
* Analyze the function $f(x) = 2x - 2x^2$ and its derivative at fixed points

3. **Analyze Convergence Rate:**
* Consider transformation $v_k = 1 - 2u_k$
* Find recurrence for v_{k+1} in terms of v_k
* Use the transformed sequence to find closed-form for v_k

4. **Translate the Condition:**
* Express $|u_k - L| \leq \frac{1}{2^{1000}}$ in terms of v_k
* Solve for the smallest k satisfying this inequality

Human: what do you think are the first few terms of the sequence and how fast does it generally converge

Model: *[Abbreviated]*
Exploring the Sequence Terms and Convergence Speed
1. **Calculating Terms:**
* Start with $u_0 = \frac{1}{4}$
* $u_1 = 2u_0 - 2u_0^2 = 2 \cdot \frac{1}{4} - 2 \cdot (\frac{1}{4})^2$
* Continue for $u_2, u_3, \ldots$

2. **Observing Convergence:**
* Compare terms to potential limits
* Analyze how $|u_k - L|$ changes with each step

3. **Relating to Transformation:**
* Calculate $v_k = 1 - 2u_k$ for first few terms
* Examine the relationship between v_{k+1} and v_k
* Determine how convergence of v_k relates to convergence of u_k

Human: it converges exponentially fast, not linearly or quadratic.

[User submits answer: 10]
[Answer is correct]

User Feedback

"it gave me general intuition how to solve it but didn't say much other than just guess and check and find a pattern."

Figure 25: Interaction between human and AI model (Gemini 2.5 Pro) on a recurrence relation problem. The model outlines a systematic approach focusing on finding limits and transformation techniques. The user identifies the exponential convergence pattern and determines the correct answer.

Minimizing Array Changes to Make Differences Equal

Human: I'm assuming this can be done pretty fast, considering we can use k as somewhat like a boundary?

Model: *[Abbreviated]*
One key insight is that you only need to ensure each pair (i, n - i - 1) has the same absolute difference. Since you can replace any element with an integer from 0 to k, you have a fixed range for potential targets.

Solution Strategy:
1. Recognize that the final array must have pairs whose difference is some integer X (<= k).
2. For each pair (i, n - i - 1), determine which X values can be achieved with 0, 1, or 2 changes.
3. Find the X that leads to the minimum total changes across all pairs.

Human: Can I have two pointers here?

Model: Yes. You could have one pointer at i = 0 and another at j = n-1, moving inward. At each step, you analyze how many replacements would be needed to make their difference match a candidate X.

Human: Can we not just count the amount of pairs with the same difference?

Model: *[Abbreviated]*
Simply counting the most common current difference can fail because you might achieve a different uniform difference with fewer total changes. You need to consider how many changes would be required for each pair to achieve each potential difference X, then choose the X requiring minimal changes overall.

Human: How do I know how many possible X's there are?

Model: All possible values of X lie in the set {0, 1, 2, ..., k}, since any replacements must be in [0, k].

Human: So this time won't be O(n)?

Model: *[Abbreviated]*
A naïve approach could lead to O(n·k) time, which is too large when n and k can both reach 10^5. Instead, for each pair, determine which differences can be achieved with 0, 1, or 2 changes, storing this information compactly. Then find the difference X that minimizes total changes.

Human: So what are we storing?

Model: For each pair, store: - The single difference achievable with 0 changes (current absolute difference) - A compact representation of differences achievable with 1 change - By default, other differences in [0, k] require 2 changes

[User attempts multiple implementations, encountering various errors]

Human: *[Final implementation attempt, which times out]*

User Feedback

"Although this AI 'seemed' smarter than the others in that it was giving me better solutions, the overall explanation was confusing such that I wasn't able to understand it! There were points where the grammar was interesting and it would throw me off. If the grammar usage was a bit better I could've implemented what it was asking me, but I started getting frustrated trying to understand what it was trying to say."

Figure 26: Interaction between human and AI model (Deepseek-V3) on a problem about minimizing array modifications to equalize differences. Despite a theoretically sound approach, communication barriers prevented successful implementation.

Finding the Largest Prime in a Complex Triangle Problem

Human: how should i approach this problem

Model: *[Abbreviated]*
Approach to the Problem
1. **Understand the Properties of z:** You have $z = a + bi$, where a and b are integers.
2. **Analyze the Modulus Condition:** Given $|z| = \sqrt{p}$ where $|z| = \sqrt{a^2 + b^2}$, we get $a^2 + b^2 = p$.
3. **Properties of p:** Consider which prime numbers can be expressed as $a^2 + b^2$.
4. **Calculate z^3:** Expand $(a + bi)^3$.
5. **Identify Real and Imaginary Parts of z^3:** Find $Re(z^3)$ and $Im(z^3)$.

Human: explain step 4 in more detail

Model: *[Abbreviated]*
You need to compute $z^3 = (a + bi)^3$ using the binomial expansion: $(a + bi)^3 = a^3 + 3a^2(bi) + 3a(bi)^2 + (bi)^3$
Remember that: - $i^2 = -1$ - $i^3 = -i$
After substituting, you can rewrite the expression and group the real and imaginary parts:
$z^3 = (a^3 - 3ab^2) + (3a^2b - b^3)i$
So $Re(z^3) = a^3 - 3ab^2$ and $Im(z^3) = 3a^2b - b^3$.
Regarding the Triangle Inequality: The condition requires that p, $Re(z^3)$, and $Im(z^3)$ form a triangle, which means: - All side lengths must be positive - Any two sides must sum to more than the third side
Human: $Re(z^3) = a^3 - 3ab^2$ and $Im(z^3) = 3a^2b - b^3$

Human: explain step 6 in more detail

Model: *[Abbreviated]*
Apply the Triangle Conditions
For a triangle with sides p, $s_2 = a^3 - 3ab^2$, and $s_3 = 3a^2b - b^3$:
1. **Positivity Conditions:** - $p > 0$ (always true since p is prime) - $s_2 > 0$, which requires $a > \sqrt{3}b$ (assuming $a, b > 0$) - $s_3 > 0$, which follows if $a > \sqrt{3}b$
2. **Triangle Inequality:** - $p + s_2 > s_3$ - $p + s_3 > s_2$ - $s_2 + s_3 > p$
These simplify to: - $|a^3 - 3a^2b - 3ab^2 + b^3| < a^2 + b^2$ - $a^3 + 3a^2b - 3ab^2 - b^3 > a^2 + b^2$
Strategy: Find pairs of positive, coprime integers (a, b) that satisfy: - a, b are positive integers - a and b are coprime - $p = a^2 + b^2$ is prime and $p < 1000$ - $a > \sqrt{3}b$ - Triangle inequality conditions hold
You've found primes 17 (from $a = 4, b = 1$), 89 (from $a = 5, b = 8$), 349 (from $a = 18, b = 5$)...
[User submits 199, which is incorrect, then 349, which is correct]

User Feedback

"Helped me with my thought process especially with certain identities and tricks. I kind of forgot certain properties of complex numbers so the AI was pretty good at getting me back up to speed. Aside from a few glitches, I feel it did a pretty good job at explaining things and setting a framework with clear steps to take."

Figure 27: Interaction between human and AI model (Gemini 2.5 Pro) on a AIME problem requiring complex number manipulation. The model provides a step-by-step approach, helping the user navigate through mathematical derivations and systematically find the largest prime meeting all conditions.