The Story is Not the Science: Execution-Grounded Evaluation of Mechanistic Interpretability Research

Xiaoyan Bai [1] **Alexander Baumgartner** [*1] **Haojia Sun** [*2] **Ari Holtzman** [1] **Chenhao Tan** [1]

Abstract

Reproducibility crises across sciences highlight the limitations of the paper-centric review system in assessing the rigor and reproducibility of research. AI agents that autonomously design and generate large volumes of research outputs exacerbate these challenges. In this work, we address the growing challenges of scalability and rigor by flipping the dynamic and developing AI agents as research evaluators. We propose the first execution-grounded evaluation framework that verifies research beyond narrative review by examining code and data alongside the paper. We use mechanistic interpretability research as a testbed, build standardized research output, and develop `MechEvalAgent`, an automated evaluation framework that assesses the *coherence* of the experimental process, the *reproducibility* of results, and the *generalizability* of findings. We show that our framework achieves above 80% agreement with human judges, identifies substantial methodological problems, and surfaces 51 additional issues that human reviewers miss. Our work demonstrates the potential of AI agents to transform research evaluation and pave the way for rigorous scientific practices.[1]

1. Introduction

Peer review has long treated the paper narrative as the primary object of evaluation (Kelly et al., 2014). While this may suffice for theoretical work, it poses challenges for empirical research, which cannot be fully assessed without executable artifacts such as code and data. Reproducibility crises across scientific fields have exposed the limitations of narrative-alone review (Collaboration, 2015; Desai et al., 2025; Youmshajekian, 2024). In one case, manipulated behavioral data resulted in a high-profile faculty termination at a leading research university (Simmons, 2024). In the context of AI research, even when code is shared, reviewers often lack the time or resources to run it, leading to a reliance on narrative descriptions that may not accurately reflect the actual implementation or results.

Agentic workflows add more challenges as they are now widely deployed in research tasks, including code assistance (Anthropic, 2025; OpenAI, 2025; Novikov et al., 2025), ideation (Zhou et al., 2024; Baek et al., 2025), and end-to-end research (Schmidgall et al., 2025; Ifargan et al., 2025). They autonomously generate large volumes of research outputs and can accelerate publication timelines, contributing to even more submissions in an already strained system. Notably, the number of papers submitted to NeurIPS increased from 1,420 submissions to 21,575 submissions from 2013 to 2025, a 15-fold increase in submission (PaperCopilot; NeurIPS PC Chairs, 2025; Desai et al., 2025). AI involvement also exacerbates the challenges of narrative-alone evaluation, as implicit hallucinations are hard to detect (Haibe-Kains et al., 2020; Beam et al., 2020).

Recent efforts have begun to use AI in the evaluation process to address this challenge, including systems that provide paper feedback (Stanford ML Group, 2025) and broader uses of LLMs or agents in peer review (Lee et al., 2025; Liang et al., 2024; Yu et al., 2024). These systems can already detect important issues such as hallucinated references (Shmatko et al., 2026), and studies show that AI generated reviews overlap well with human reviews (Liang et al., 2024). However, most approaches examine the coherence and consistency of the story in the paper and ignore all the executable resources. In other words, the focus is on how appealing the story is rather than whether the story is true.[2]

In this work, we seek to verify the science itself, not just the story told about it. Motivated by this view, we propose a general evaluation framework that combines narrative and execution to reveal errors that narrative-alone review misses,

[1]University of Chicago [2]Carnegie Mellon University. Correspondence to: Xiaoyan Bai <smallyan@uchicago.edu>.

Preprint. February 10, 2026.

[1]Code available: https://github.com/ChicagoHAI/MechEvalAgent/

[2]Narrative-alone evaluation is even more concerning in AI review, where authors have inserted prompts in papers to raise the scores of AI judges (Else, 2025).

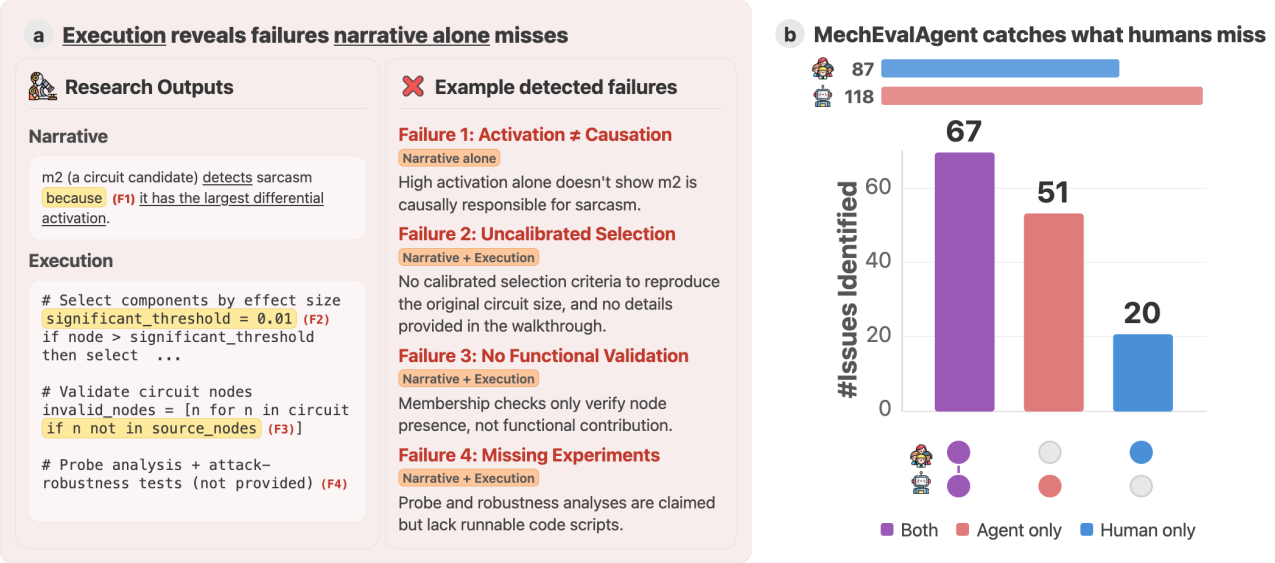


Figure 1. (a) Execution-grouned evaluation uncovers failures that narrative-alone review misses. In this example, Failures 2, 3, and 4 require execution beyond narrative review. (b) As a highlight of our results, we find that `MechEvalAgent` surfaces 51 additional issues that human reviewers overlooked.

as shown in Figure 1, demonstrating how agents can support rather than undermine scientific review.

We propose the first execution-grounded evaluation framework that standardizes research outputs by bundling execution resources with narrative. This enables systematic assessment of *coherence* of the experiment process, *reproducibility* of the results, and *generalizability* of the findings beyond what the final paper alone can provide.

We build `MechEvalAgent`, an automated evaluation agent that implements our pipeline. We focus on mechanistic interpretability as a testbed because its claims are often expressed in terms of concrete mechanisms and interventions that can be directly tested by rerunning experiments and applying the proposed analyses to new models or inputs. The field also follows relatively standardized methodological patterns, which makes failures in execution and evaluation easier to isolate. Finally, generalizability is a central open question in mechanistic interpretability, making it a natural setting to test whether evaluation can go beyond narrative. While we instantiate our framework in mechanistic interpretability, the framework itself is not domain-specific and can be adapted to other areas of scientific research.

We evaluate `MechEvalAgent` on 30 research outputs in mechanistic interpretability. `MechEvalAgent` achieves above 80% agreement with human experts. As shown in Figure 1b, it captures most failures identified by humans (67 of 87) and surfaces 51 *additional* issues that humans missed. Many of these issues illustrated in Figure 1a require execution to detect, such as missing data selection criteria that only emerge during reproduction (Failure 2), validation code that checks list membership rather than task validity (Failure 3), or missing files that prevent reproducing key results (Failure 4). Removing code access or execution substantially reduces agreement to about 45% on average, and `MechEvalAgent` evaluates faster than humans, who take 2.2 hours per task on average, aligned with research in other disciplines (Abogunrin et al., 2025). These results highlight the value of execution-grounded evaluation.

To summarize, our main contributions are as follows:

- We develop the first execution-grounded evaluation framework for research outputs that goes beyond narrative-alone review by standardizing research outputs and building systematic evaluation suites.
- We build `MechEvalAgent` to instantiate this framework in mechanistic interpretability and evaluate 30 research outputs from both humans and AI research agents.
- Our evaluation results align with human experts and surface 51 issues overlooked by human reviewers, demonstrating the importance of combining narrative and execution-grounded assessments.

2. Method

`MechEvalAgent` expects research outputs to include both narrative and execution resources. We standardize research outputs and introduce an evaluation framework based on these outputs to enable execution-grounded assessment.

Unified research outputs. We propose a unified standard for research outputs. Research outputs are expected to include two components:

- **Narrative**: For human-written papers, we extract the the

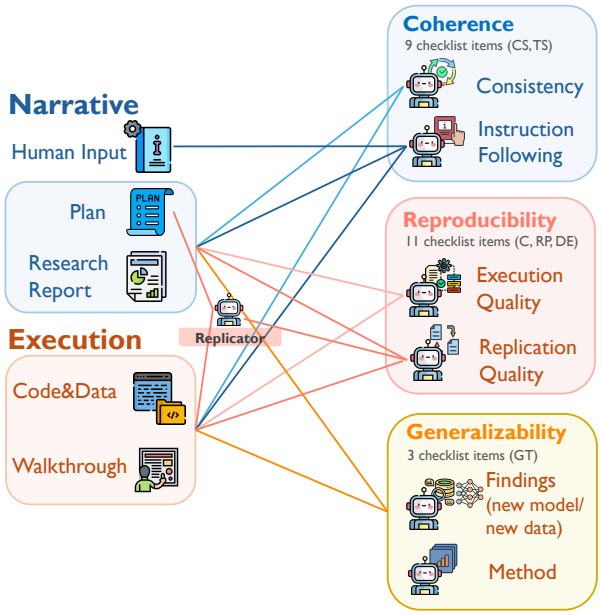


Figure 2. Overview of the `MechEvalAgent` framework. Research outputs are evaluated on coherence, reproducibility, and generalization, with each sub-dimension handled by an agent that takes in the relevant inputs.

plan and the *report*. For agent-generated outputs, we expect the research trace beyond a final paper, including *human prompts*, for instance. The plan specifies the goal, hypothesis, constraints, and intended methodology. Comparing it against the executed code enables checks of plan-implementation consistency. This helps detect goal drift and cases where an intended methodology is quietly replaced by a different method that yields favorable results. The report summarizes goals, methods, results, and conclusions, links claims to evidence, and can briefly discuss future directions and implications.

- **Execution Resources**: We expect the *implementation*, including code and data, together with a *walkthrough* of it. This enables grounded checks of runnability and correctness, and surfaces failures that are invisible in narrative-alone review, such as broken environments, API mismatches, or incorrect metric computations. The walkthrough also allows us to assess whether sufficient information is provided for reproduction.

These resources make it possible to verify not only *what* an agent claims, but also *how* the claim was produced. For example, a report may state that an intervention improves a metric, but execution can reveal that the code does not run, implements a different computation than described, or fails to reproduce the reported numbers.

Our proposed evaluation framework. Our evaluation framework formalizes the review process into explicit checks. When evaluating research outputs, a reviewer typically wants to know three things: (i) whether the claims are consistent with the provided evidence, (ii) whether the results can be reproduced, and (iii) whether the findings generalize beyond the original setting. As illustrated in Figure 2, our `MechEvalAgent` operationalizes these questions in the following three components, with more detailed description in Appendix A.

Coherence. Coherence evaluates on *consistency* and *instruction following*. It asks whether the research outputs are internally consistent and aligned with its stated goals, including whether the implementation and reported results support the claims and whether the artifact follows the intended research objective.

Reproducibility. Reproducibility evaluates on *execution quality* and *replication quality*. It asks whether the data is presented, the code is runnable and correctly implements the described computations, and whether an independent replication run can reproduce the reported results. We forbid the access to report to avoid hallucination. As shown in Figure 2, we have another agent to evaluate on the replication quality including check whether there is external reference and hallucination in replication and whether the result and the conclusion matches with the original ones.

Generalizability. In this part, an agent reads the research repository and checks whether the reported findings generalize beyond the original setting, including to a new model, new data instances, or related tasks. This evaluation process measures whether one can learn something from the research by testing the predictions based on the findings in a new situation, considered an important component of generalizability (Kim et al., 2025). This evaluation is well-suited to mechanistic interpretability, where research often identifies mechanisms in a specific model and task while requiring generalizable insights across architectures or contexts. Therefore, whether the agent is able to do this serves as an indirect measure of whether the original research captured generalizable insight rather than task- or model-specific patterns.

In practice, we evaluate each component through structured binary checklists. The advantage is that checklists reduce subjectivity and enable consistent aggregation of evaluation results across agents and tasks and enables comparisons with human experts.[3] Figure 2 summarizes the checklist structure and overall workflow. Each sub-dimension is implemented by a dedicated agent that evaluates multiple items in the checklist. For example, our checklists cover statistical significance (`CS5`), effect size (`CS3`), and justification (`CS4`) for *coherence*, code runnability (`C1`) for *reproducibil-*

[3]Anecdotally, we find that `MechEvalAgent` can identify even more issues when the checklist is not provided, as it can explore more aspects of the research output. However, for systematic evaluation and comparison, we focus on checklist-based evaluation in this work.

ity, and generalization to new models and data instances (`GT1, GT2`) for *generalizability*.[4] The full list of checklist items is included in Table 2, and additional implementation details and example outputs are included in Appendix A.1.

Building `MechEvalAgent`. We develop `MechEvalAgent` on Claude Code, with execution and logging support provided by Scribe (Goodfire AI, 2025). Scribe integrates code generation with Jupyter notebooks, making execution traces and errors explicit. This enables reliable execution checking and qualitative analysis. Specifically, we reorganize both narrative and execution resources, including input prompts to the research agent, plans, code implementations, relevant data, walkthroughs, and final reports. The structure for human-written repositories differs slightly and is explained in Section 3. We introduce specialized agents responsible for evaluating different metrics as discussed above. During early iterations, we identified a risk of `MechEvalAgent` modifying source files. To prevent this, we enforce strict file access restrictions and integrate with GitHub for version control monitoring to ensure no unintended modifications occur. To guard against hallucination in reproduction, we route replication results through a separate verification agent that checks result fidelity. With this modular, execution-grounded design, `MechEvalAgent` can be extended to other domains beyond mechanistic interpretability. Figure 2 illustrates the detailed input structure of `MechEvalAgent`.[5]

3. Experiment Setup

To cover a diverse range of research scenarios, we evaluate our framework and `MechEvalAgent` on three types of research outputs, with ten examples each. The selected tasks are shown in Table 3:

- **Replication of Research.** The agent is given a concrete research goal and hypothesis from an established paper and must generate a plan and implement experiments that directly test it. This setting tests instruction following in a realistic workflow where a human delegates a well-defined experiment. It is also prone to hallucinated reasoning, since the agent may rely on memorized conclusions rather than grounded execution. Our `MechEvalAgent` surfaces such failures by checking whether the implementation and results actually support the stated hypothesis.
- **Open-ended Research Questions.** Agents are given open-ended questions with only high-level goals (e.g., identifying sarcasm-related circuits), without a known ground truth. This setting reflects exploratory research, where agents must form their own hypotheses and outcomes can be either positive or negative. Coherence is especially important at this stage, since an agent should be able to report null or negative results rather than defaulting to overly optimistic conclusions.
- **Human-Written Repositories.** Our evaluation is not limited to agent-generated research. Human-written repositories can exhibit similar issues, such as overclaiming or poorly reproducible code. Since these repositories often lack explicit plan files, we extract the goal, claims, and methodology from the accompanying paper to apply our pipeline in a unified manner. In this setting, we evaluate all aspects of the pipeline except instruction following, which is not applicable.

Research agents naturally produce richer outputs than traditional artifacts, including plans, intermediate logs, and human prompts. We fully utilize and standardize these into a unified format for systematic checklist evaluation, enabling checks on plan-implementation consistency and verification that conclusions follow from execution. Our research agent is built on Scribe, which processes our instruction prompts and structures the research output according to our standardized format. We include an example of the detailed prompt in Appendix A.2.

Evaluating our evaluation framework. To evaluate our evaluation framework and `MechEvalAgent`, we first measure **agreement** between human evaluators and `MechEvalAgent`. Human evaluators are asked to use the exact same checklists and instructions as the agents to assess the research *independently*, including reproducing the repository and testing generalizability. We measure the agreement between human and agent judgments. This evaluates whether our agents produce evaluations that are consistent with human reasoning. We also examine evaluator **efficiency** by measuring the time humans spent on evaluation. In addition, we evaluate the **rated quality** of the agent's evaluation output. Human experts assess whether the issues reported by our agents are correct, meaningful, and complete by scoring their assessment with the agents from 1 to 5 (strongly disagree - strongly agree with the agent's judgement). Due to limited resources, we have three human authors as annotators.

We run each automated evaluation three times. We use AND logic for PASS (equivalent to OR logic for FAIL), where a task is marked as PASS only when all runs return PASS. We additionally analyze majority vote and cross-run stability in Appendix B. Due to limited resources, we asked three human experts to evaluate 30 research generations, with each generation evaluated by exactly one expert. Each expert also evaluated the quality of the automated evaluation output for a single evaluation run.

[4] Checklist item of statistical significance is captured by top ML conference, as well as reproducibility of the main results and broader impact of the research.

[5] Please refer to supplementary materials for detailed input templates.

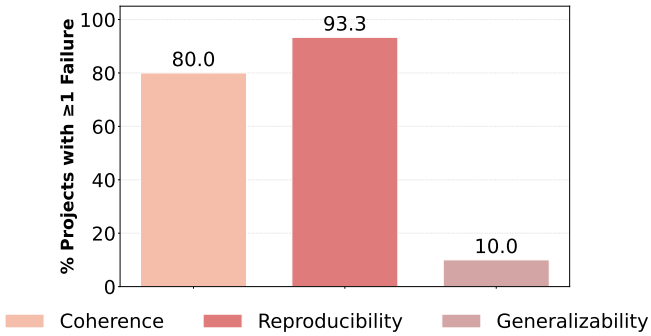


Figure 3. Percentage of projects with at least one failure per dimension. Over 90% of tasks fail in reproducibility, and 80% fail in coherence.

Ablation study. To isolate the value of execution, we compare `MechEvalAgent` to two ablated evaluators. The *Doc-Only* evaluator reads only the final report and cannot inspect or run code, approximating limited paper-only review. The *No-Execution* evaluator sees the full repository but cannot execute it, so reproducibility and generalizability are only judged from written evidence rather than executions. More details of the setting are shown in Appendix A.2.

4. Results

Our results show that execution failures and narrative weaknesses are common across tasks, undermining reliability. `MechEvalAgent` aligns with human judgments while surfacing additional issues that humans miss. Ablations further confirm the value of execution-grounded evaluation.

4.1. An Overview of Key Findings

Execution and narrative issues are prevalent in research outputs (Figure 3). Over 90% of the tasks have at least one failure in reproducibility, largely driven by execution errors, and 80% of the tasks fail in coherence, mainly due to lack of consistency, as shown in Figure 3 and Figure 8. Execution failures often stem from model loading, shape calculations, and environment setup, and they appear in both agent-generated and human-written repositories. Our checklists also surface narrative weaknesses in evidential support, such as unreliable effect sizes, weak statistical significance, or conclusions based on insufficient results.

`MechEvalAgent` aligns with human judgments, while execution helps surface additional issues beyond narrative. As shown in Figure 4, rated quality of the evaluation outputs by `MechEvalAgent` is high, with scores above 4.7 out of 5 across all dimensions. The agreement is above 80% across all dimensions (Figure 6). Figure 1 shows that `MechEvalAgent` captures most failures identified by humans (67 of 87) and surfaces 51 additional issues. As shown in Figure 5, most of the additional failures are concentrated in reproducibility and generalizability, which are more time-consuming for humans to evaluate. This pattern suggests execution enables `MechEvalAgent` to uncover more issues, especially execution ones.

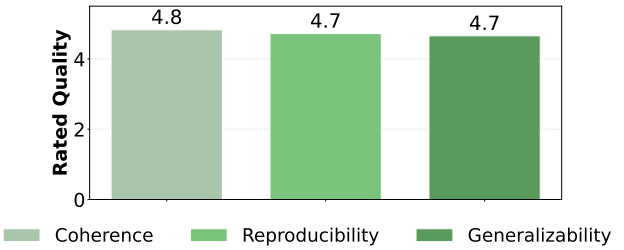


Figure 4. Human-rated quality on `MechEvalAgent` evaluations (1-5 Likert scale, 1 = Strongly Disagree, 5 = Strongly Agree). All dimensions show ratings above 4.7, indicating high quality of agent assessments.

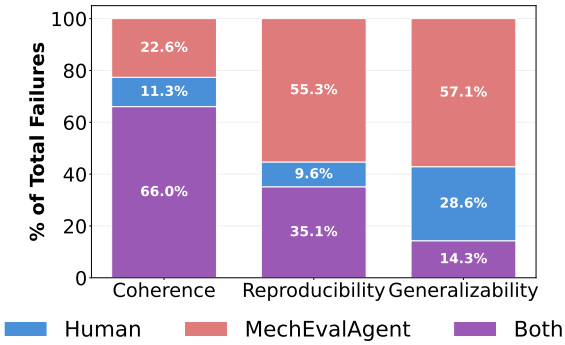


Figure 5. Failure breakdown comparing human-identified and agent-identified issues. `MechEvalAgent` surfaces more unique issues in all three dimensions.

Ablation highlights the problem of narrative-alone evaluation (Figure 6). Both ablated evaluators agree with humans far less than the full `MechEvalAgent` pipeline, especially on execution dependent dimensions. This indicates that an AI evaluator without execution access and limited to narrative review is less reliable. In these settings, the evaluator also tends to over-assign FAIL as shown in Figure 8, suggesting that execution provides needed grounding for calibrated judgments.

4.2. A Closer Look at Failures and Disagreements

We now closely look into common failures identified by `MechEvalAgent` and human experts, and analyze disagreements between them.

Execution failures and weak reproducibility remain common. Zooming in the high reproducibility failures observed in Figure 3, Figure 7 shows that execution failures are more common. The high failure rate in execution evaluation reflects difficulties with missing packages and transformer internals, often showing up as shape errors in residual streams and attention heads (e.g., "head-level activation patching mismatch" from a replication task, and "API compatibil-

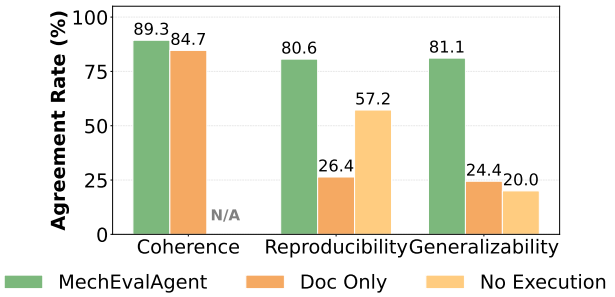


Figure 6. Agreement with human experts across the full `MechEvalAgent` pipeline and two ablated variants (Doc-Only and No-Execution). Full `MechEvalAgent` pipeline shows high agreement across all dimensions. Both ablations perform substantially worse than the full pipeline.

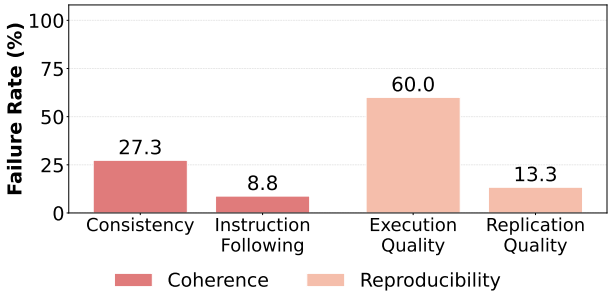


Figure 7. Average failure rates by subgroup. Consistency checks fail 27.3% on average, and execution quality checks fail 60% of the time.

ity issue" from a human repository). In agent-generated research, these errors often appear early and are fixed in subsequent blocks. For example, after encountering a shape mismatch or missing key, the agent may inspect tensor shapes or enumerate available keys to find the correct one.

However, a more serious problem is that results are sometimes not reproducible, undermining confidence in the conclusions. In `acronyms`, a replication task that seeks a mechanistic account of predicting multiple consecutive tokens (Appendix A.2), the research agent's report claims to use the logit correlation metric to evaluate the circuit. However, `MechEvalAgent` found that during replication, though the identified circuits overlap, logit correlation metrics deviate by more than 8% from the original (Original value: 0.66, Replication value: 0.72), and this discrepancy appears across all three evaluation runs. Although the replicated results have better performance than the original, it reveals concerns in validity of the original method and the evaluation procedure. Inspecting the cause shown in Table 1, we found that the research agent evaluated correlation on only a subset of examples without explicitly noting this choice (first 20 examples), whereas `MechEvalAgent` followed the stated instructions and evaluated on the full dataset. This observation point to issues in the original evaluation procedure and raise concerns about reproducibility.

We observe a related issue in `erasing` (Gandikota et al., 2024), a human-written repository that proposes an approach to concept-level unlearning, with more detailed description in Appendix A.2. Here, `MechEvalAgent` found that probe-training code and adversarial testing scripts are missing. Although the work primarily focuses on effective unlearning, the sections on probe and activation analysis and robustness to attacks serve as important evaluation components. While this failure is captured by our checklist item on consistency between stated experiments and implemented code, it also directly limits reproducibility because a user cannot reconstruct the full set of evaluation. Beyond missing evaluation scripts, `MechEvalAgent` found that `belief` (Prakash et al., 2025), another human-written repository that studies how LMs represent characters' beliefs (Appendix A.2), contains an invalid Jupyter notebook file related to their BigToM causal model experiments.

Narratives of agent-generated research fail to be consistent and lack sufficient justification. With respect to *coherence* failures, Figure 7 shows that the average failure rate in consistency subgroups can reach 27.3%. As shown in Figure 8, the most common consistency failure is missing statistical significance. We also observe failure rates of 23.3% for plan-implementation consistency (`CS2`) and 6.7% for effect size (`CS3`).

A particularly concerning failure is the lack of sufficient justification which has 23.3% failure rate (Figure 8). For example, as shown in Table 1, on the `ioi` task, which is a replication task focus on mechanistically understand the indirect object identification task (Appendix A.2), `MechEvalAgent` noted that a "strongly supports" conclusion was drawn despite a -4.2% circuit performance, which contradicts the reported evidence. These failures suggest potential hallucination in the research agent, possibly due to memorization of existing work in training data.

`MechEvalAgent` surfaces issues humans overlook. Table 1 shows that `MechEvalAgent` tends to provide more detailed and specific rationales than human experts, making the feedback more actionable for improving research agents. Human rationales are typically more general, likely due to limited time for thorough inspection. In `unanswerable`, an open-ended question task that explains how models solve unanswerable questions (Appendix A.2), `MechEvalAgent` flagged that the choice of final circuit size lacks explicit justification and that validation results showing circuit ablation being only 0.73× as impactful as random ablation directly contradict the claimed importance. In contrast, human judges only noted that the justification was insufficient.

Figure 5 further shows that `MechEvalAgent` surfaces more unique issues across all dimensions: 22.6% in coherence, 55.3% in reproducibility, and 57.1% in generalizability. While some differences arise from interpre-

Table 1. Examples of issues identified by `MechEvalAgent` (🤖) and human experts (🧑‍🔬). `MechEvalAgent` provides specific, execution-grounded rationales, while human rationales tend to be more general. **R** = research agent generated repository, **H** = human-written repository. The detailed description of the tasks are in Appendix A.2.

Code or Claims in Repository	Issues Identified
[R] `for ex in examples[:20]: #code for calculating the logic correlation metric` [Source: `acronyms`: to understand multiple consecutive token predictions]	🤖 Logit correlation metrics deviate by more than 8% during evaluation. (`DE1`) 🧑‍🔬 N/A
[R] `logic_diff_retain = circuit_diffs.mean() / baseline_diff *100` logic_diff = -4.2% [Source: `ioi`: to understand indirect object identification]	🤖 The circuit verification shows -4.2% performance retention. (`CS4`) 🧑‍🔬 The justification of getting the result is insufficient
[H] No related code presented in the repository. In their paper, they reported probe analysis (Section:Probing and Activation Analysis) and tested the robustness of the attack (Section: Robustness to Attacks). [Source: `erasing` (Gandikota et al., 2024) Accessed: Jan 27, 2026]	🤖 2 experiments are missing: (1) no attack code exists despite results being reported. (2) no probing or activation analysis code. (`CS2`) 🧑‍🔬 There are parts of the results that we don't see implemented
[H] This notebook file has syntax error and is an invalid notebook. [Source: `belief` (Prakash et al., 2025) Accessed: Jan 27, 2026]	🤖 `causalmodel_exps.ipynb` file is not runnable (`C1`) 🧑‍🔬 `causalmode_exps.ipynb` could not be opened because it doesn't finish closing the curly braces of the json file.

tation, the agent consistently identifies execution-related problems. For example, as discussed earlier, in `acronyms`, `MechEvalAgent` found that the logit correlation metric deviates substantially, whereas human judges did not identify any issues.

Human evaluation can also be affected by limited resources or domain knowledge, and `MechEvalAgent` helps compensate for these constraints. In `uncertainty`, an open-ended question task that aims to identify circuits related to uncertainty (Appendix A.2), human judges criticized the agent under `CS4` for switching to GPT-2 Medium without justification. However, code inspection reveals an `if` condition that switches models only after a load failure. By executing and inspecting the code, `MechEvalAgent` provides a more accurate assessment.

Conversely, the 20 issues identified exclusively by human experts show that human evaluation still adds valuable insights beyond `MechEvalAgent`. For example, in `multilingual`, an open-ended question task that aims to understand how models "think" under multilingual instructions (Appendix A.2), human judges identified a hallucination where the reported support value of $+0.13$ for an identified neuron was actually derived only from the single translation task, rather than from the average performance across the full task.

Disagreements stem from different interpretations of checklists. Beyond surfacing issues, `MechEvalAgent` shows high agreement with human evaluators, including both issues and pass cases. However, there are still some disagreements, especially in reproducibility and generalizability (Figure 6). Figure 10a shows that much of this comes from the Redundancy check (`C3`). Our checklist specifies that code adding new information should not be considered redundant. However, when the agent copies the same block repeatedly to run on different inputs instead of writing reusable functions, our evaluators mark it as redundant while human experts do not. This disagreement highlights inefficiencies in agent code generation, which is common among coding agents (Horikawa et al., 2025; Azeem et al., 2025). The high rated quality on this evaluation also supports that the evaluation is reasonable.

Generalizability metrics also show disagreement. For method generalizability (`GT3`), humans and evaluators differ on what counts as a new method. For model generalizability (`GT1`), disagreement arises from evaluation depth. Due to different model sizes, evaluation sometimes only assess high-level generalization. For example, the evaluation may simply check whether the function of early layers transfers to a new model, which human experts consider insufficient to demonstrate true generalization. Therefore, their rated quality falls mostly into Agree category.

Efficiency comparison. In terms of efficiency, to perform agreement evaluation, we asked human judges to use the exact same checklists and instructions as the agents to assess the research, including reproducing the repository and testing generalizability. They spent an average of 2.2 hours per evaluation task, with evaluations of human-written repositories often exceeding 3.35 hours, as shown in Table 4. In contrast, `MechEvalAgent` typically completes evaluations in under 30 minutes for agent generated tasks and around one hour for human-written repositories.

Overall, `MechEvalAgent` raises the floor of evaluation quality by efficiently surfacing issues that humans may overlook, while human expertise remains valuable for catching

edge cases that automated evaluation misses.

5. Related Work

Agents for Autonomous Research. LLM-based agents now automate research across multiple stages. Some focus on ideation and hypothesis generation (Zhou et al., 2024; Baek et al., 2025; Gottweis et al., 2025), others on execution-grounded discovery through iterative code and environment feedback (Novikov et al., 2025; Jiang et al., 2025). End-to-end systems close the loop from ideation to experimentation and writing, producing papers and repositories with code (Lu et al., 2024; Yamada et al., 2025; Schmidgall et al., 2025; Ifargan et al., 2025; Jansen et al., 2025). However, these systems produce heterogeneous outputs, complicating comparison. Our work standardizes research agent outputs into a unified trace that enables evaluation beyond narrative review alone.

Evaluating Research. Recent efforts evaluate research agents along different dimensions. LLM-based peer review assesses paper narratives: Liang et al. (2024) found GPT-4 reviews overlap with human reviews at rates comparable to inter-reviewer agreement, and Zhuang et al. (2025) survey LLM capabilities for checklist verification and error detection. However, these approaches treat code as readable but not executable. Benchmarks have begun addressing this gap: ResearchRubrics (Sharma et al., 2025b) and DeepResearch Bench (Du et al., 2025) assess long-form written outputs, while Exp-Bench (Kon et al., 2025) extends evaluation to code execution and experimental correctness. Other efforts benchmark ideation and hypothesis generation (Guo et al., 2025; Liu et al., 2025). While these approaches advance grounded evaluation, they largely focus on reproducing a given experiment, with only access to the paper, which is unfair replication tasks. Our execution-grounded assessment framework is the first framework to evaluate this external validity, while also verifying that conclusions are grounded in actual execution rather than narrative alone.

6. Concluding Discussion

Our work aims to verify the science beyond reviewing the paper. To do that, we propose a novel evaluation framework consisting of coherence, reproducibility, and generalizability. We further develops `MechEvalAgent` and uses mechanistic interpretability as a testbed to compare agent-based evaluation with human evaluation. Our approach achieves above 80% agreement with human experts while surfacing 51 additional issues that humans miss. Our results demonstrate that AI agents can raise the floor of evaluating scientific research by exposing concrete, execution-grounded issues that humans often overlook.

From Narrative to Execution. We advocate a shift from narrative-alone review toward integrating execution-grounded evaluation into research assessment to improve rigor and reliability in scientific research. Execution is crucial for research evaluation, yet reviewers often skip it due to time constraints and incomplete replication instructions. Our results point toward a natural division of labor. Human reviewers excel at judging novelty, contribution, and framing, areas where AI is 10 times less likely to comment (Liang et al., 2024). The 20 issues identified exclusively by humans in our study involve interpretation and scope judgments requiring broader context. Automated evaluation, by contrast, excels at execution-heavy checks that humans skip under deadline pressure: verifying that code runs, that outputs match claims, and that results replicate. These are precisely the areas where `MechEvalAgent` surfaces issues that humans miss.

Passing `MechEvalAgent` does not guarantee high quality, but failing it highlights concrete, actionable weaknesses. Moreover, `MechEvalAgent` pinpoints what needs revision and provide specific, execution-grounded rationales, while human judges often give general rationales like "insufficiently justified". This aligns with BaHammam (2025), who argue that AI should support human editors by catching basic errors before external review.

Beyond evaluation, `MechEvalAgent`'s feedback can help identify hallucinations in research agents and improve their reliability. As discussed in Section 4.2, research agents often report positive results despite weak evidence and exploit ambiguity to make their outputs appear successful. Execution-grounded evaluation catches these issues early, before they propagate into published claims.

In short, we call for continued efforts to build evaluation frameworks that integrate narrative review with execution-grounded checks, moving research assessment toward verifying the science itself rather than just the story.

Challenges in Agent-Based Evaluation. To proceed, we note three particular challenges. First, binary checklists can be brittle when evidence is mixed. They force hard decisions where graded judgments would better reflect uncertainty, making outcomes sensitive to borderline cases and underspecified scenarios. Because research is highly open-ended, this underspecified space may be larger than in other domains. Second, we observe occasional optimistic behavior in ambiguous settings. In early runs, the evaluation agent sometimes modifies source files to justify a PASS despite prompts forbidding edits, or downplays negative evidence under underspecified failure handling. Although introducing stricter prompts nearly mitigates these issues, they highlight the need for caution in agent-based evaluation. More broadly, evaluation agents may default to neutral or positive judgments when interpretation is unclear. Binary checklists expose this tendency by forcing categorical

decisions, motivating clearer guidance for handling uncertainty in automated evaluation. The third challenge arises from limited instruction-following ability. As noted above, even when we explicitly forbid modifications to source files, the agent sometimes still makes changes. When evaluating code quality, the model is instructed to copy and re-run the exact code, yet it occasionally executes specific functions instead or silently alters the copied code. A related issue arises in metric definitions. For example, although the prompt clearly states that fixing a previous error should not count as redundancy, the model sometimes marks such cases as redundant, even when its own rationale acknowledges the change fixes an earlier error. These behaviors highlight limitations in instruction following that remain a challenge for coding agents. Despite these challenges, `MechEvalAgent` achieves above 80% agreement with human evaluators, demonstrating the viability of automated research evaluation. We encourage continued efforts to improve instruction following and uncertainty handling in evaluation agents.

Limitations. Our current implementation uses Claude Code as the sole evaluation agent. While our results demonstrate strong alignment with human judgment, future work could explore ensembles of different models to further improve robustness. Additionally, due to resource constraints, each repository was reviewed by a single human expert rather than multiple independent reviewers, and human evaluation was conducted on the first evaluation run rather than all three runs. Despite these constraints, the high agreement between our pipeline and human experts suggests that our findings are reliable, and we expect that additional human reviewers would further validate the patterns we observe.

Acknowledgements

We gratefully thank Yonatan Belinkov, Tal Haklay, Austin Kozlowski, Jiachen Liu, and Tamar Rott Shaham for fruitful discussions. We also thank Modal for generously providing computing credits.

Impact Statement

Our work proposes a new evaluation framework that combines narrative review with execution-grounded checks to make research evaluation more rigorous. More broadly, our work aims to advance research evaluation toward more scalable and automated processes. This effort is not limited to machine learning and can be generalized to other fields. This research does not present specific ethical concerns or societal implications beyond those.

References

Abogunrin, S., Muir, J. M., Zerbini, C., and Sarri, G. How much can we save by applying artificial intelligence in evidence synthesis? results from a pragmatic review to quantify workload efficiencies and cost savings. *Frontiers in Pharmacology*, 16:1454245, 2025.

Anthropic. Claude Code: an agentic coding assistant. https://code.claude.com/docs/en/sub-agents, 2025. Agentic coding workflows and subagent framework for automated code generation and execution.

Azeem, S., Naveed, M. S., Sajid, M., and Ali, I. Ai vs. human programmers: Complexity and performance in code generation. *VAWKUM Transactions on Computer Sciences*, 13(1):201–216, 2025.

Baek, J., Jauhar, S. K., Cucerzan, S., and Hwang, S. J. Researchagent: Iterative research idea generation over scientific literature with large language models. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 6709–6738, 2025.

BaHammam, A. S. Peer review in the artificial intelligence era: A call for developing responsible integration guidelines, 2025.

Beam, A. L., Manrai, A. K., and Ghassemi, M. Challenges to the reproducibility of machine learning models in health care. *Jama*, 323(4):305–306, 2020.

Camerer, C. F., Dreber, A., Holzmeister, F., Ho, T.-H., Huber, J., Johannesson, M., Kirchler, M., et al. Evaluating the replicability of social science experiments in nature and science between 2010 and 2015. *Nature Human Behaviour*, 2(9):637–644, 2018.

Collaboration, O. S. Estimating the reproducibility of psychological science. *Science*, 349(6251):aac4716, 2015.

Desai, A., Abdelhamid, M., and Padalkar, N. R. What is reproducibility in artificial intelligence and machine learning research? *AI Magazine*, 46(2):e70004, 2025.

Du, M., Xu, B., Zhu, C., Wang, X., and Mao, Z. Deepresearch bench: A comprehensive benchmark for deep research agents, 2025. URL https://arxiv.org/abs/2506.11763.

Else, H. Ai is transforming peer review — and that's raising concerns. *Nature*, 2025. doi: 10.1038/d41586-025-02172-y. URL https://www.nature.com/articles/d41586-025-02172-y.

Errington, T. M., Mathur, M., Soderberg, C. K., Denis, A., Perfito, N., Iorns, E., and Nosek, B. A. Investigating the replicability of preclinical cancer biology. *eLife*, 10: e71601, 2021.

Feucht, S., Wallace, B., and Bau, D. Vector arithmetic in concept and token subspaces. In *Second Mechanistic Interpretability Workshop at NeurIPS*, 2025. URL https://arithmetic.baulab.info.

Gandikota, R., Feucht, S., Marks, S., and Bau, D. Erasing conceptual knowledge from language models. *arXiv preprint arXiv:2410.02760*, 2024.

García-Carrasco, J., Maté, A., and Trujillo, J. C. How does gpt-2 predict acronyms? extracting and understanding a circuit via mechanistic interpretability. In *International Conference on Artificial Intelligence and Statistics*, pp. 3322–3330. PMLR, 2024.

Goodfire AI. Scribe: Jupyter Server + Notebooks for CLI Agents, 2025. URL https://github.com/goodfire-ai/scribe. GitHub repository; gives CLI agents access to Jupyter servers and automatically records code and outputs in notebooks.

Gottweis, J., Weng, W.-H., Daryin, A., Tu, T., Palepu, A., Sirkovic, P., Myaskovsky, A., Weissenberger, F., Rong, K., Tanno, R., et al. Towards an ai co-scientist. *arXiv preprint arXiv:2502.18864*, 2025.

Gross, J., Agrawal, R., Kwa, T., Ong, E., Yip, C. H., Gibson, A., Noubir, S., and Chan, L. Compact proofs of model performance via mechanistic interpretability. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Guo, S., Shariatmadari, A. H., Xiong, G., Huang, A., Kim, M., Williams, C. M., Bekiranov, S., and Zhang, A. Ideabench: Benchmarking large language models for research idea generation. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pp. 5888–5899, 2025.

Gurnee, W., Horsley, T., Guo, Z. C., Kheirkhah, T. R., Sun, Q., Hathaway, W., Nanda, N., and Bertsimas, D. Universal neurons in gpt2 language models. *arXiv preprint arXiv:2401.12181*, 2024.

Haibe-Kains, B., Adam, G. A., Hosny, A., Khodakarami, F., of Directors Shraddha Thakkar 35 Kusko Rebecca 36 Sansone Susanna-Assunta 37 Tong Weida 35 Wolfinger Russ D. 38 Mason Christopher E. 39 Jones Wendell 40 Dopazo Joaquin 41 Furlanello Cesare 42, M. A. Q. C. M. S. B., Waldron, L., Wang, B., McIntosh, C., Goldenberg, A., Kundaje, A., et al. Transparency and reproducibility in artificial intelligence. *Nature*, 586(7829):E14–E16, 2020.

Hanna, M., Liu, O., and Variengien, A. How does GPT-2 compute greater-than?: Interpreting mathematical abilities in a pre-trained language model. *arXiv preprint arXiv:2305.00586*, 2023.

Heimersheim, S. and Janiak, J. A circuit for python docstrings in a 4-layer attention-only transformer. Alignment Forum, 2023. URL https://tinyurl.com/ycynk5kv.

Hernandez, E., Sharma, A. S., Haklay, T., Meng, K., Wattenberg, M., Andreas, J., Belinkov, Y., and Bau, D. Linearity of relation decoding in transformer language models. In *Proceedings of the 2024 International Conference on Learning Representations*, 2024.

Horikawa, K., Li, H., Kashiwa, Y., Adams, B., Iida, H., and Hassan, A. E. Agentic refactoring: An empirical study of ai coding agents. *arXiv preprint arXiv:2511.04824*, 2025.

Ifargan, T., Hafner, L., Kern, M., Alcalay, O., and Kishony, R. Autonomous llm-driven research—from data to human-verifiable research papers. *NEJM AI*, 2(1): AIoa2400555, 2025.

Jansen, P., Tafjord, O., Radensky, M., Siangliulue, P., Hope, T., Dalvi, B., Majumder, B. P., Weld, D. S., and Clark, P. Codescientist: End-to-end semi-automated scientific discovery with code-based experimentation. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 13370–13467, 2025.

Jiang, Z., Schmidt, D., Srikanth, D., Xu, D., Kaplan, I., Jacenko, D., and Wu, Y. Aide: Ai-driven exploration in the space of code. *arXiv preprint arXiv:2502.13138*, 2025.

Kelly, J., Sadeghieh, T., and Adeli, K. Peer review in scientific publications: benefits, critiques, & a survival guide. *Ejifcc*, 25(3):227, 2014.

Kim, B., Hewitt, J., Nanda, N., Fiedel, N., and Tafjord, O. Because we have llms, we can and should pursue agentic interpretability, 2025. URL https://arxiv.org/abs/2506.12152.

Kon, P. T. J., Liu, J., Zhu, X., Ding, Q., Peng, J., Xing, J., Huang, Y., Qiu, Y., Srinivasa, J., Lee, M., Chowdhury, M., Zaharia, M., and Chen, A. Exp-bench: Can ai conduct ai research experiments?, 2025. URL https://arxiv.org/abs/2505.24785.

Kozlov, A. Persona collapse. https://github.com/akozlo/Persona-Collapse-blog, 2025. Accessed: 2026-01-21.

Lazic, S. E. Internal replication as a tool for evaluating reproducibility in preclinical experiments, 2025. URL https://arxiv.org/abs/2506.03468.

Lee, J., Lee, J., and Yoo, J.-J. The role of large language models in the peer-review process: opportunities and challenges for medical journal reviewers and editors. *Journal of Educational Evaluation for Health Professions*, 22, 2025.

Liang, W., Zhang, Y., Cao, H., Wang, B., Ding, D. Y., Yang, X., Vodrahalli, K., He, S., Smith, D. S., Yin, Y., et al. Can large language models provide useful feedback on research papers? a large-scale empirical analysis. *NEJM AI*, 1(8):AIoa2400196, 2024.

Liu, H., Huang, S., Hu, J., Zhou, Y., and Tan, C. Hypobench: Towards systematic and principled benchmarking for hypothesis generation. *arXiv preprint arXiv:2504.11524*, 2025.

Lu, C., Lu, C., Lange, R. T., Foerster, J., Clune, J., and Ha, D. The ai scientist: Towards fully automated open-ended scientific discovery. *arXiv preprint arXiv:2408.06292*, 2024.

Mathwin, C., Corlouer, G., Kran, E., Barez, F., and Nanda, N. Identifying a preliminary circuit for predicting gendered pronouns in gpt-2 small. MATS/Apart Mechanistic Interpretability Hackathon, 2023. URL https://itch.io/jam/mechint/rate/1889871.

McDougall, C. Balanced bracket classifier: Mechanistic interpretability tutorial. ARENA 3.0 Curriculum, 2023. URL https://arena3-chapter1-transformer-interp.streamlit.app/[1.5.1]_Balanced_Bracket_Classifier. Educational materials with documented circuit solution.

McDougall, C., Conmy, A., Rushing, C., McGrath, T., and Nanda, N. Copy suppression: Comprehensively understanding an attention head. *arXiv preprint arXiv:2310.04625*, 2023.

Meng, K., Bau, D., Andonian, A., and Belinkov, Y. Locating and editing factual associations in GPT. *Advances in Neural Information Processing Systems*, 35, 2022.

Nanda, N., Chan, L., Lieberum, T., Smith, J., and Steinhardt, J. Progress measures for grokking via mechanistic interpretability. In *International Conference on Learning Representations*, 2023.

NeurIPS PC Chairs. Reflections on the 2025 review process from the program committee chairs, 2025. URL https://tinyurl.com/mw48e9rs. Accessed: 2026-01-27.

Novikov, A., Vũ, N., Eisenberger, M., Dupont, E., Huang, P.-S., Wagner, A. Z., Shirobokov, S., Kozlovskii, B., Ruiz, F. J., Mehrabian, A., et al. Alphaevolve: A coding agent for scientific and algorithmic discovery. *arXiv preprint arXiv:2506.13131*, 2025.

Olsson, C., Elhage, N., Nanda, N., Joseph, N., DasSarma, N., Henighan, T., Mann, B., Askell, A., Bai, Y., Chen, A., et al. In-context learning and induction heads. *Transformer Circuits Thread*, 2022.

OpenAI. OpenAI Codex: cloud-based software engineering agent. https://openai.com/index/introducing-codex/, 2025. Software engineering agent that can write, test, and deploy code autonomously.

PaperCopilot. Neurips statistics. URL https://papercopilot.com/statistics/neurips-statistics/. Accessed: 2026-01-27.

Prakash, N., Shapira, N., Sharma, A. S., Riedl, C., Belinkov, Y., Shaham, T. R., Bau, D., and Geiger, A. Language models use lookbacks to track beliefs, 2025. URL https://arxiv.org/abs/2505.14685.

Sandmann, E., Lapuschkin, S., and Samek, W. Iterative inference in a chess-playing neural network, 2025. URL https://arxiv.org/abs/2508.21380.

Schmidgall, S., Su, Y., Wang, Z., Sun, X., Wu, J., Yu, X., Liu, J., Liu, Z., and Barsoum, E. Agent laboratory: Using llm agents as research assistants. *arXiv preprint arXiv:2501.04227*, 2025.

Sharma, A. S., Rogers, G., Shapira, N., and Bau, D. Llms process lists with general filter heads. 2025a.

Sharma, M., Zhang, C. B. C., Bandi, C., Wang, C., Aich, A., Nghiem, H., Rabbani, T., Htet, Y., Jang, B., Basu, S., Balwani, A., Peskoff, D., Ayestaran, M., Hendryx, S. M., Kenstler, B., and Liu, B. Researchrubrics: A benchmark of prompts and rubrics for evaluating deep research agents, 2025b. URL https://arxiv.org/abs/2511.07685.

Shmatko, N., Adam, A., and Esau, P. Gptzero finds 100 new hallucinations in neurips 2025 accepted papers. Online, January 21 2026. URL https://tinyurl.com/4n2h9jfk. Accessed: 2026-01-23.

Simmons, J. Harvard's gino report reveals how a dataset was altered, July 9 2024. URL https://datacolada.org/118. Accessed: 2026-01-25.

Stanford ML Group. Paperreview.ai: Stanford agentic reviewer for ai-assisted paper feedback. Online, 2025. URL https://paperreview.ai/. Accessed: 2026-01-23.

Tan, L., Huang, K.-W., Shi, J., and Wu, K. Interpdetect: Interpretable signals for detecting hallucinations in retrieval-augmented generation. *arXiv preprint arXiv:2510.21538*, 2025.

Todd, E., Li, M. L., Sharma, A. S., Mueller, A., Wallace, B. C., and Bau, D. Function vectors in large language models. In *Proceedings of the 2024 International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=AwyxtyMwaG. arXiv:2310.15213.

Wang, K., Variengien, A., Conmy, A., Shlegeris, B., and Steinhardt, J. Interpretability in the wild: a circuit for indirect object identification in GPT-2 small. In *International Conference on Learning Representations*, 2023.

Yamada, Y., Lange, R. T., Lu, C., Hu, S., Lu, C., Foerster, J., Clune, J., and Ha, D. The ai scientist-v2: Workshop-level automated scientific discovery via agentic tree search. *arXiv preprint arXiv:2504.08066*, 2025.

Youmshajekian, L. Exclusive: Psychology researcher loses phd after allegedly using husband in study and making up data, April 26 2024. URL https://tinyurl.com/32tcj86c. Accessed: 2026-01-25.

Yu, H., Hong, Z., Cheng, Z., Zhu, K., Xuan, K., Yao, J., Feng, T., and You, J. Researchtown: Simulator of human research community. *arXiv preprint arXiv:2412.17767*, 2024.

Zhou, Y., Liu, H., Srivastava, T., Mei, H., and Tan, C. Hypothesis generation with large language models. *arXiv preprint arXiv:2404.04326*, 2024.

Zhuang, Z., Chen, J., Xu, H., Jiang, Y., and Lin, J. Large language models for automated scholarly paper review: A survey. *Information Fusion*, 124:103332, 2025.

A. Design and Experiment Setup Details

In this section, we further discuss the details about the three dimensions we pick: coherence, reproducibility, and generalizability. Then, we provide the details of the items in the checklist and the tasks we pick.

A.1. Design Details

This appendix provides additional details for the three evaluation dimensions used in `MechEvalAgent`. The main paper presents a concise overview in Section 2.

Coherence. Coherence evaluation includes *consistency evaluation* and *instruction following* evaluation. *Consistency evaluation* checks whether the agent's implementation, results, and conclusions match one another. It tests hallucinated findings by verifying that the implementation adheres to the stated plan, runs successfully, and produces the outputs referenced in the documentation. It also checks effect size, justification, and statistical significance. Together, these checks ensure that the reasoning chain from plan to code to analysis is logically sound and grounded in executable evidence. Besides, we also evaluate *instruction following*. This step assesses whether the agent performed the intended experiment. `MechEvalAgent` has extra access to human input prompts to check whether it pursues the specified hypotheses, respects constraints, and follows the prescribed setup.

Reproducibility. We evaluate two aspects of reproducibility: *execution quality* and *replication quality*. For execution quality, we check code runnability. However, runnable code is necessary but not sufficient for reproducibility, as it may still produce incorrect results due to over-claiming or hallucination.

For replication quality, we test whether the same model, in a fresh session and without access to the original report, can reproduce the reported results. This mirrors standard scientific practice, where reproducibility is established by independently re-running experiments (Errington et al., 2021; Lazic, 2025; Camerer et al., 2018). We forbid access to the report to prevent implicit hallucination, where the model might copy conclusions regardless of actual results or reverse-engineer the replication from reported findings. The plan and implementation should contain all information needed to replicate. A separate replicator evaluator then verifies whether the same conclusions are reached. To further guard against hallucination, we check for references to external sources or reliance on memorized information not present in the repository. Successful replication provides stronger evidence than runnability alone, indicating that the experiment was well-specified, sufficiently documented, and free of hidden dependencies.

Generalization. In this stage, the agent reads the research repository and evaluates whether the reported findings generalize beyond the original setting, such as to a new model, new data instances, or related tasks. The agent is allowed up to three trials to identify suitable alternative models or data. This constraint prevents unbounded search while still giving the agent enough flexibility to explore and produce stable, informative feedback.

A.2. Experiment Setup Details.

Checklist. Across all settings, we use structured binary checklists to evaluate coherence, reproducibility, and generalizability, as shown in Table 2. Each evaluator produces checklist-level judgments with concise rationales, along with a more detailed analysis report. The evaluation process, including execution and logs, is stored in a Jupyter notebook. Then it will generate a separate json file to summarize the results and the rationale. An example `MechEvalAgent` output of consistency evaluation on `ioi` task is shown below:


```
"Checklist": {
  "CS1_Results_vs_Conclusion": "FAIL",
  "CS2_Plan_vs_Implementation": "PASS",
  "CS3_Effect_Size": "FAIL",
  "CS4_Justification": "FAIL",
  "CS5_Statistical_Significance": "FAIL"
},
"Rationale": {
  "CS1_Results_vs_Conclusion": "The documentation claims the circuit
    'strongly supports' the hypothesis, but the actual circuit
    verification shows negative performance (-4.2% of baseline).",
```

```
    "CS2_Plan_vs_Implementation": "All five steps in the plan were
      implemented. While verification results were poor, the plan
      steps were executed as specified.",
    "CS3_Effect_Size": "The circuit verification shows -4.2% performance
      retention, meaning the circuit performs worse than random.",
    "CS4_Justification": "Multiple key choices lack justification:
      attention thresholds are arbitrary; the 'strongly supports'
      conclusion contradicts the -4.2% circuit performance.",
    "CS5_Statistical_Significance": "No error bars, confidence intervals,
      or statistical tests are reported."
  }
```

Example Input Prompt for Research Agents. Our research agent is built in Scribe and we design specific prompts to guide them through the tasks and regulate the output. In replication tasks, we give it more detailed research task, experiment setup, hypothesis to test, and possible method to use. While in open-ended tasks, we only give it the general questions and possible methodology. Here is an illustration of our input prompts. For more detailed prompts, please refer to supplementary material.

```
You are a **senior mechanistic interpretability researcher**.
### MODEL AND DATA ...
### GOAL
Identify a **precise circuit**|
a subset of attention heads and MLPs|that reproduces the model's
**sarcasm recognition behavior**.
The focus is on **how the model internally resolves conflict between
literal content and intended meaning**.
...
### TASK DESCRIPTION
Sentences in the sarcasm dataset typically contain **conflicting cues**
between surface meaning and pragmatic intent.
Example:...
Phenomena of interest may include (non-exhaustive):...
...
### SRC_NODES, CONSTRAINTS, EXPECTED OUTPUTS, FILES TO PRODUCE,
DOCUMENTATION REQUIREMENTS, OUTPUT SUMMARY
...
```

Task Design. As we introduced in Section 3, we design three different categories of tasks including replication tasks, open-ended question tasks, and research done by human researchers. Replication tasks request the research agent to replicate an existing work, while open-ended questions are proposed by us and have not had specific research conducted on them. The research repositories of replication and open-ended tasks are all generated by an agent. In replication tasks, our input prompt to the research agent specifies the task, the hypothesis, and possible methodology (but not limited to those) they can use. In open-ended questions, our input prompt specifies the research question we are interested in and potential methods they can use to approach the question. But the agents are required to come up with their own hypothesis and method. The details of our task choices are shown in Table 3. We provide additional details on each task category below.

Replication Tasks. The replication tasks comprise ten well-established benchmarks from mechanistic interpretability literature, each with documented ground-truth circuits that enable systematic evaluation.

`ioi` (Indirect Object Identification) (Wang et al., 2023) requires predicting the indirect object in sentences with an "ABB" pattern (e.g., "When John and Mary went to the store, John gave a drink to..." $\rightarrow$ "Mary"). This task isolates a 26-head circuit with seven functional classes, including Name Mover heads and S-Inhibition heads. `acronyms` (Acronym prediction) (García-Carrasco et al., 2024) tests multi-token generation by predicting three-letter acronyms from expanded forms (e.g., "Chief Executive Officer" $\rightarrow$ "CEO"). `greater_than` (Greater-than) (Hanna et al., 2023) evaluates numerical reasoning with sentences like "The war lasted from 1732 to 17..." where the model must predict a valid year greater than the starting year. `pronoun` (Gendered pronoun resolution) (Mathwin et al., 2023) requires resolving pronouns using

gender information (e.g., "The nurse sent the doctor a request because... [she]"). `copy` (Copy suppression) (McDougall et al., 2023) investigates a negative mechanism where attention heads attend to previous token instances to suppress repetition. `induction` (Induction heads) (Olsson et al., 2022) represent a foundational benchmark: given "A B ... A", predict "B". The circuit involves composition between Previous Token heads and Induction heads, demonstrating Q-K composition. `modular` (Modular addition) (Nanda et al., 2023) requires reverse-engineering a transformer computing $(a + b) \mod p$. The circuit employs discrete Fourier transform features and trigonometric identities, testing whether agents can identify mathematical structure. `docstring` (Docstring completion) (Heimersheim & Janiak, 2023) tests code understanding on a 4-layer attention-only transformer: given a function definition, the model predicts argument names in the docstring. `balanced_bracket` (Balanced bracket classification) (McDougall, 2023) classifies parenthesis sequences as balanced or unbalanced, testing whether agents can discover state-tracking mechanisms analogous to a counter. `max_of_k` (Max-of-K) (Gross et al., 2024) requires predicting the maximum value from a list of integers.

Open-ended Research Questions. The open-ended tasks comprise ten exploratory research questions without established ground-truth circuits, testing whether agents can formulate coherent hypotheses and conduct rigorous investigations when outcomes are uncertain. These questions span linguistic phenomena, model limitations, and abstract reasoning capabilities.

`sarcasm` asks how sarcastic intent is represented, whether the model maintains separate representations for literal versus intended meaning, and how context modulates interpretation. `multilingual` explores how models handle instructions in different languages, probing whether internal processing is language-agnostic or maintains language-specific pathways. `typo` investigates how models map misspelled tokens to intended forms, testing whether error correction uses explicit circuits or emerges from distributional robustness. `unanswerable` probes how models recognize queries that cannot be answered from available context, testing for circuits that detect answerability. `uncertainty` asks whether dedicated circuits encode model confidence or epistemic uncertainty, distinct from prediction content. `count` investigates why models struggle to generate exactly n items, exploring whether counting relies on approximate mechanisms that degrade with sequence length. `inevitability` tests whether GPT-2 distinguishes semantic inevitability (physical causation, e.g., "She dropped the glass. It hit the floor and...") from narrative inevitability (story logic, e.g., "The detective found the final clue. The mystery was..."). `irreversibility` probes which layers encode whether events can be undone (e.g., "He closed the door" vs. "He broke the vase"). `persona` (Kozlov, 2025) investigates why different assigned personas converge on identical preferences, probing whether persona representations are shallow or deeply integrated. `moral` asks whether models separate moral valence (whether an action is wrong) from consequentialist evaluation, testing for dissociable ethical representations.

Human-Written Repositories. The human-written repositories comprise ten published research projects from the mechanistic interpretability literature, enabling evaluation of our pipeline on artifacts produced through standard academic workflows rather than agent generation. These repositories vary in scope, methodology, and documentation quality, providing a realistic test of generalization.

`filter` (Sharma et al., 2025a) identifies attention heads that implement general list-filtering operations across diverse tasks. `universal` (Universal neurons) (Gurnee et al., 2024) catalogs neurons in GPT-2 that activate consistently across contexts, proposing a taxonomy of interpretable features. `function_vector` (Todd et al., 2024) demonstrates that in-context learning can be captured by single vectors that, when added to activations, induce task performance without exemplars. `rome` (Meng et al., 2022) locates factual associations in GPT and introduces Rank-One Model Editing for targeted knowledge modification. `relation` (Relation decoding) (Hernandez et al., 2024) shows that transformer language models decode relational knowledge through approximately linear maps from subject representations to object predictions. `erasing` (Concept erasing) (Gandikota et al., 2024) proposes methods for removing specific concepts from language model representations, with applications to unlearning. `belief` (Belief tracking) (Prakash et al., 2025) investigates how language models represent characters' beliefs in theory-of-mind tasks, identifying "lookback" mechanisms that track belief states. `interpdetect` (Tan et al., 2025) develops interpretable signals for detecting hallucinations in retrieval-augmented generation for financial QA. `leela` (Chess reasoning) (Sandmann et al., 2025) analyzes iterative inference in Leela Chess Zero, probing how neural networks refine position evaluations through computation. `arithmetic` (Vector arithmetic) (Feucht et al., 2025) examines whether concept vectors support algebraic operations in both concept and token subspaces, testing compositionality of learned representations.

Ablation Study Settings. Doc-Only ablation removes access to all artifacts except the final documentation or report. The evaluator cannot see the research plan, code, or execution traces. All judgments are based solely on the written narrative. Consequently, it can assess only textual coherence and reported generalization, while reproducibility is inferred from descriptions

rather than verified. Since some human-written repositories include code snippets in their documentation, we also include code evaluation. Specifically, we evaluate execution quality(C1–C4), consistency(CS1–CS5), generalizability(GT1–GT3), and replication quality(RP1–RP3).

No-Execution ablation provides the evaluator with the full repository, including the plan, code, walkthrough, and report, but disallows code execution. Reproducibility and generalization are evaluated based solely on written evidence. Since instruction following and consistency evaluation do not involve code execution in our original pipeline, we exclude these metrics from this ablation. Specifically, we evaluate execution quality(C1–C4), generalizability(GT1–GT3), and replication quality(RP1–RP3). Comparing this ablation to our full pipeline isolates the value of execution-grounded evaluation, highlighting failures that cannot be detected through code inspection alone.

Table 2. Structured binary checklist. Please refer to supplementary material for the complete description for each item.

Dimensions	Aspects	Checklists
COHERENCE	Consistency Evaluation	**CS1:** All evaluable conclusions in the documentation match the results originally recorded in the notebook. **CS2:** A plan file exists, and all steps in the final version of the plan are reflected in the implementation. **CS3:** The reported effects have a clearly non-trivial magnitude (effect size) relative to baseline behavior or variability, such that the conclusions do not rely on marginal or negligible changes. **CS4:** All key design choices and intermediate conclusions are explicitly justified, explaining why each design was chosen and how each conclusion is supported. **CS5:** Key experimental results supporting the main claims report appropriate measures of uncertainty or significance (e.g., error bars, confidence intervals, or statistical tests), with a clear explanation of what variability they capture.
	Instruction Following	**TS1:** The goal described in the plan file matches the input stated goal. **TS2:** The goal described in the plan file matches the input stated goal. **TS3:** The plan file's methodology follows the input intended direction and covers the required analyses. **TS4:** For every circuit component identified, the tests confirm that its behavior matches the hypothesized function described in the given plan.
REPRODUCIBILITY	Execution Quality	**C1:** The block executes without error. **C2:** The logic implements the described computation correctly (indexing, metric formulas, patching logic, dataset handling). **C3:** The block duplicates another block's computation without adding new information. (Revising previous wrong results does not considered as redundant.) **C4:** The block does not contribute to achieving the project goal as defined in the lan, code walkthrough, or documentation.
	Replication Quality	**RP1:** The experiment can be reconstructed from the plan and code-walk without missing steps or required inference beyond ambiguous interpretation. **RP2:** The environment (packages, models, data) can be restored and run without unresolved version or dependency issues. **RP3:** Replicated results are stable across multiple runs. **RP4:** **(only when demo exists)** pass when all of the following conditions are satisfied: (1) The demo can be executed or followed without referencing hidden or external materials. (2) Experiment or result claimed in the original paper / plan is can be demonstrated in the demo. (3) The demo specifies all required inputs, configurations, and execution steps needed to reproduce the demonstrated results. **DE1:** Replicated documentation reports results (metrics, trends, qualitative findings) that match the original documentation within acceptable tolerance (within 5% deviation). **DE2:** The replicated documentation presents conclusions and interpretations consistent with the original. **DE3:** No new information appears that is absent from or unsupported by the original documentation.
GENERALIZABILITY	Finding Generalizability	**GT1:** The newly-proposed concept is predictable on a new model, and can be verified through at least one example. **GT2:** The newly-proposed concept is predictable on a new data instance, and can be verified through at least one example.
	Method Generalizability	**GT3:** If the work propose a new method, the method can be applied to another similar task, and can be verified through at lease one example.

Table 3. Task suite used to evaluate research agents across replication, open-ended questions, and human-authored research.

Task Category	Task Description
REPLICATION	`ioi`: Identify the circuit for predicting indirect objects in sentences (Wang et al., 2023). `acronyms`: Find components for predicting multiple consecutive tokens in acronym completions (García-Carrasco et al., 2024). `greater_than`: Locate the circuit that determines if a year exceeds a previous year (Hanna et al., 2023). `docstring`: Find the circuit that predicts function argument names in Python docstrings (Heimersheim & Janiak, 2023). `pronoun`: Identify how the model resolves pronouns to their referent entities (Mathwin et al., 2023). `copy`: Investigate how copy suppression heads prevent repeated token predictions (McDougall et al., 2023). `induction`: Explain the mechanism behind decreased probability for recently-seen tokens (Olsson et al., 2022). `modular`: Reverse-engineer a model trained to compute modular addition (Nanda et al., 2023). `balanced_bracket`: Understand the circuit for classifying balanced brackets (McDougall, 2023). `max_of_k`: Locate the circuit that identifies the maximum value in a list (Gross et al., 2024).
OPEN-ENDED	`sarcasm`: Identify how the model represents and detects sarcasm. `multilingual`: Investigate how the model processes multilingual instructions. `unanswerable`: Understand how the model handles unanswerable questions. `uncertainty`: Determine whether a dedicated circuit exists for representing uncertainty. `typo`: Explain how the model maps misspelled tokens to correct forms. `persona`: Investigate why different personas converge to similar preferences (Kozlov, 2025). `inevitability`: Distinguish how the model represents semantic vs. narrative inevitability. `irreversibility`: Find which layers encode the irreversibility of events. `moral`: Separate the model's representations of moral wrongness from bad outcomes. `count`: Understand why the model struggles to generate a specified number of words.
HUMAN REPO	`filter`: Identify general filter heads that process list elements (Sharma et al., 2025a). `interpdetect`: Detect hallucination signals in financial question answering (Tan et al., 2025). `arithmetic`: Study vector arithmetic in concept and token subspaces (Feucht et al., 2025). `universal`: Identify neurons with consistent functions across contexts (Gurnee et al., 2024). `leela`: Analyze iterative inference in a chess-playing neural network (Sandmann et al., 2025). `relation`: Test linearity of relation decoding in transformer language models (Hernandez et al., 2024). `function_vector`: Study how in-context learning is encoded as function vectors (Todd et al., 2024). `erasing`: Evaluate methods for removing conceptual knowledge from models (Gandikota et al., 2024). `belief`: Investigate how models track beliefs using lookback mechanisms (Prakash et al., 2025). `rome`: Locate and edit factual associations in GPT models (Meng et al., 2022).

Table 4. Average evaluation time per task by category. Human-written repositories require the longest evaluation time due to their larger codebases and more complex documentation.

Task Category	Average Time
Replication Tasks	1.0 hr
Open-ended Questions	2.3 hr
Human-written Repositories	3.35 hr

B. Detailed Results

Detailed Failure Rates. Figure 8 shows the failure rate averaged across all tasks for each metric. `MechEvalAgent` shows high failure rates in statistical significance and execution quality. The No-Execution and Doc-Only baselines exhibit even higher failure rates, especially in generalizability and replication quality where execution is crucial. However, agreement with human judgments is low for these baselines, indicating over-assignment of failures.

Figure 9 shows pass rates for each checklist item broken down by task category. Overall pass rates are similar across categories. On average, human-written repositories achieve 78.9%, replication tasks achieve 75.0%, and open-ended tasks 75.7%. Code quality metrics (`C1-C4`) show consistently high failure rates. Statistical significance (`CS5`) shows 100% failure rate in both replication and open-ended tasks, while human-written repositories achieve 60% pass rate. Replication tasks show relatively higher pass rates on instruction following (`TS1-TS4`), as expected given their well-defined experimental goals. Open-ended tasks show more variation in coherence metrics, reflecting the greater ambiguity in evaluating exploratory research.

Agreement and Rated Quality. We measure whether `MechEvalAgent` verdicts match human evaluations using the same checklist (agreement), and how much human experts agree with the rationales `MechEvalAgent` provides (rated quality). As shown in Figure 10, `MechEvalAgent` achieves high agreement with human experts across all task categories: 86.4% for replication tasks, 80.5% for open-ended tasks, and 80.5% for human-written repositories. Consistency metrics (`CS1-CS5`) show particularly strong agreement, reaching 90-100% across most categories. Reproducibility metrics (`RP1-RP3`) also achieve high agreement rates of 90-100%. The lowest agreement occurs in `C3` (Redundancy), with only 30-40% agreement across categories, and `GT3` (method generalizability), which shows 30% agreement in replication tasks. Those failures are likely due to different interpretation of the checklist as we discussed in Section 4.2. Figure 11 shows average rated quality of `MechEvalAgent` evaluations, which is above 4 (Agree) across all metrics.

Beyond agreement with human evaluation, `MechEvalAgent` also identifies unique issues that humans miss, as discussed in Section 4.2. On average, each project has 1.57 issues in coherence, 2.83 issues in reproducibility, and 0.17 issues in generalizability that are uniquely identified by `MechEvalAgent`, as shown in Figure 12. Reproducibility is the dimension where most unique issues are found. Fewer issues are found in generalizability, primarily due to our task design. As shown in Figure 9, all generalizability failures occur in human-written repositories. This is partially because our replication and open-ended tasks are not designed for specific architectures, whereas human-written repositories often contain demonstration code tailored to particular architectures. Additionally, passing the generalizability checklist does not guarantee that results transfer to other models or that the methodology is stable across settings, as discussed in Section 4.2.

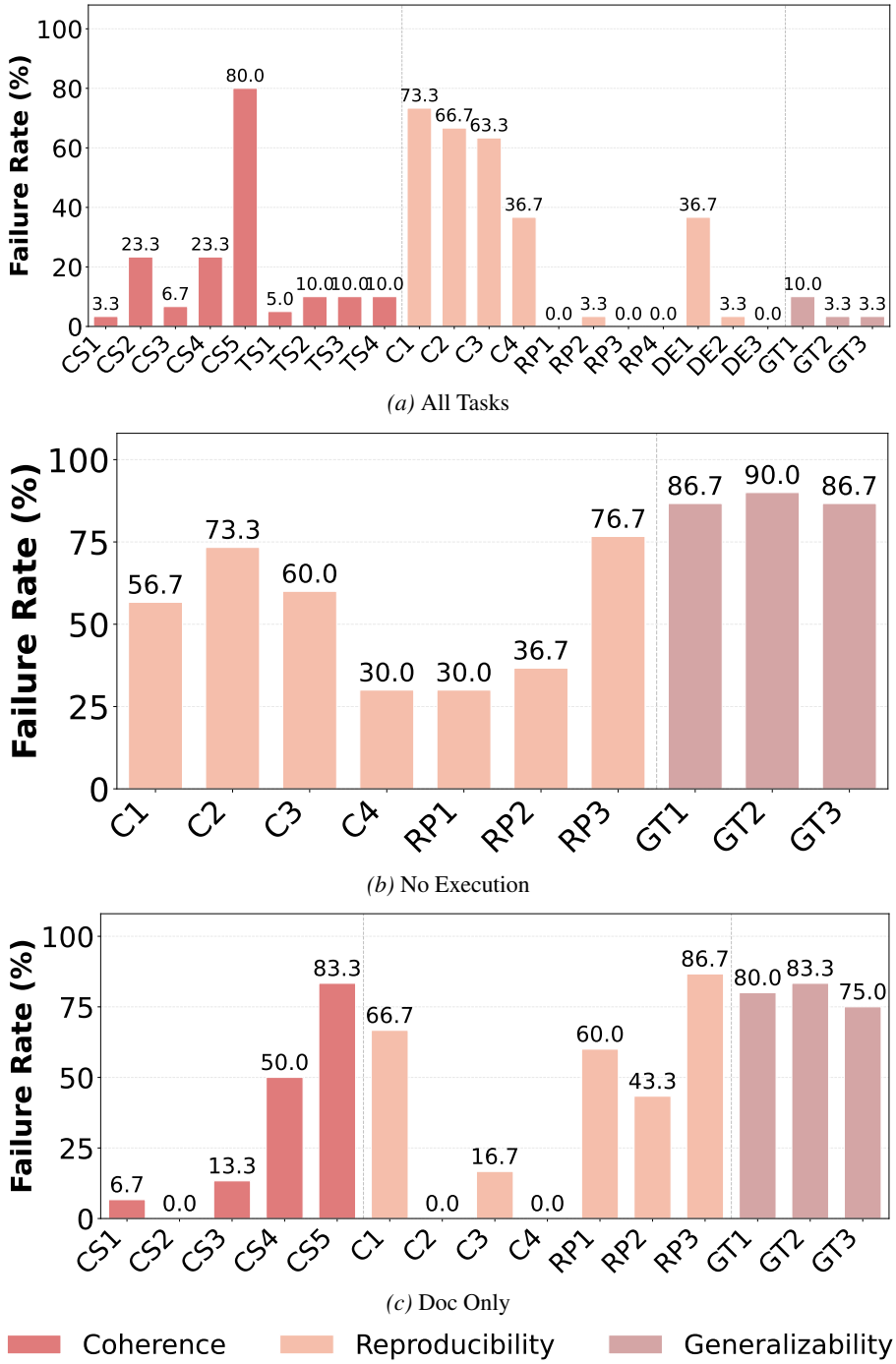
Efficiency Comparison. Table 4 shows the average evaluation time per task across different categories. Replication tasks require the shortest evaluation time (1.0 hour on average), as they have well-defined experimental setups and smaller codebases. Open-ended questions take longer (2.3 hours) due to the need to understand the hypothesis and method proposed by research agents. Human-written repositories require the most time (3.35 hours), reflecting their larger codebases, more complex documentation, and the additional effort needed to extract goals and methodology from accompanying papers.

Stability. We run evaluation for three times for each research project. Here, we measure the stability by checking the agreement across three runs. In Figure 13 and Figure 14, we show the proportion of perfect agreement (3 runs are the same) and one-dissent agreement (1 run is different).

Our evaluations in consistency, instruction following, generalizability, and reproduction are mostly stable, as shown in Figure 13a. We attribute the instability to two main factors. First, our binary checklists cannot enumerate every possible situation, particularly given the diverse input space of research artifacts, leading to inconsistent interpretations across runs. Second, we observe an interesting behavioral pattern in Claude Code, which is a reluctance to provide negative feedback.

This can contribute to the instability during evaluation. The evaluation agent tends to exploit ambiguities in the checklist to assign PASS, which we discuss further in Section 6. As discussed in Section 4.2, different interpretations of the checklist also contribute to the instability. Despite these sources of instability, agreement with human evaluators remains high, and the instability itself serves as a useful signal, highlighting cases that warrant closer human inspection.

To show more determined failures, we also apply majority voting to failures, where a task is marked as FAIL when there are two or more runs return FAIL. Issues identified through majority voting represent more consistent failures that persist across multiple evaluation runs. As shown in Figure 15, majority voting yields an overall higher pass rate. The most common failures remain in `CS5` (statistical significance) with 80% failure rate, followed by failures in execution quality. Agreement with human judges improves to 89.4% when using majority voting.

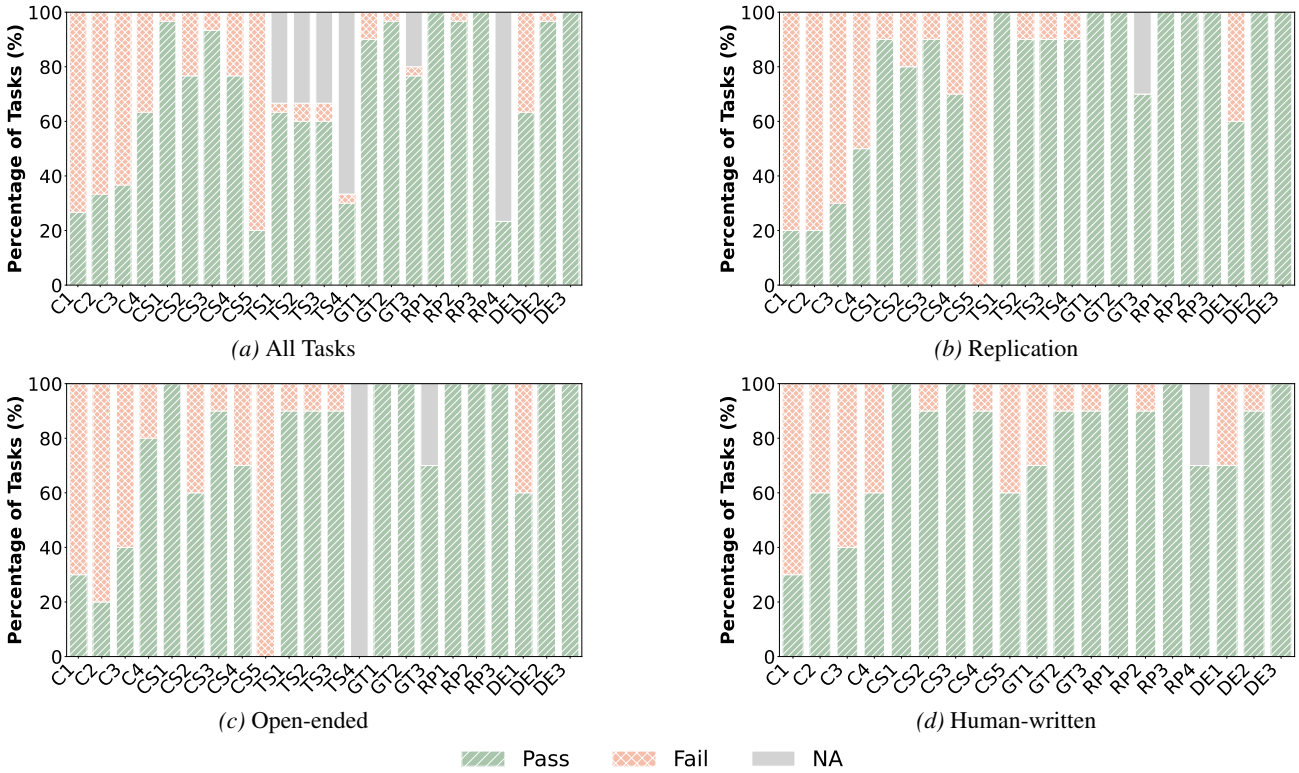


(a) All Tasks

(b) No Execution

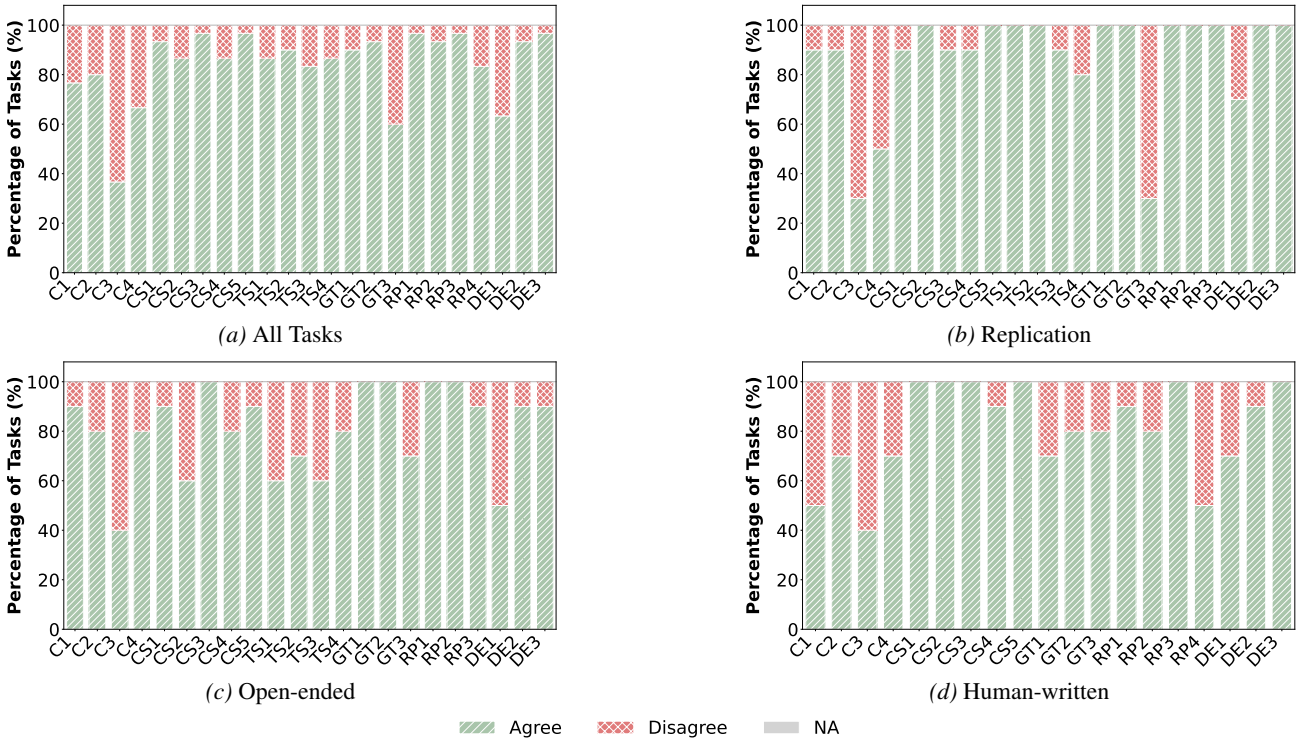
(c) Doc Only

Figure 8. Average failure rate of all the tasks with `MechEvalAgent`, Doc-only, and No-Execution. Under `MechEvalAgent` evaluations, statistical significance(`CS5`) achieve 80% failure rate. The failure rate in execution quality(`C1-C4`) and replicated results fidelity (`DE1`) are high. In contrast, FAIL is over-assigned in the Doc-only and No-Execution settings, especially in generalizability.



(a) All Tasks *(b)* Replication *(c)* Open-ended *(d)* Human-written

Figure 9. Pass rates using AND logic. (a) All 30 tasks combined. (b) Replication tasks. (c) Open-ended tasks. (d) Human-written repositories. Code quality metrics (C1-C4) and CS5 (result justification) show highest failure rates.



(a) All Tasks *(b)* Replication *(c)* Open-ended *(d)* Human-written

Figure 10. Agreement between `MechEvalAgent` and human experts. Bars show percentage of tasks where agent's evaluation matched human assessment. (a) All tasks (above 80% average). (b) Replication tasks. (c) Open-ended tasks. (d) Human-written repositories.

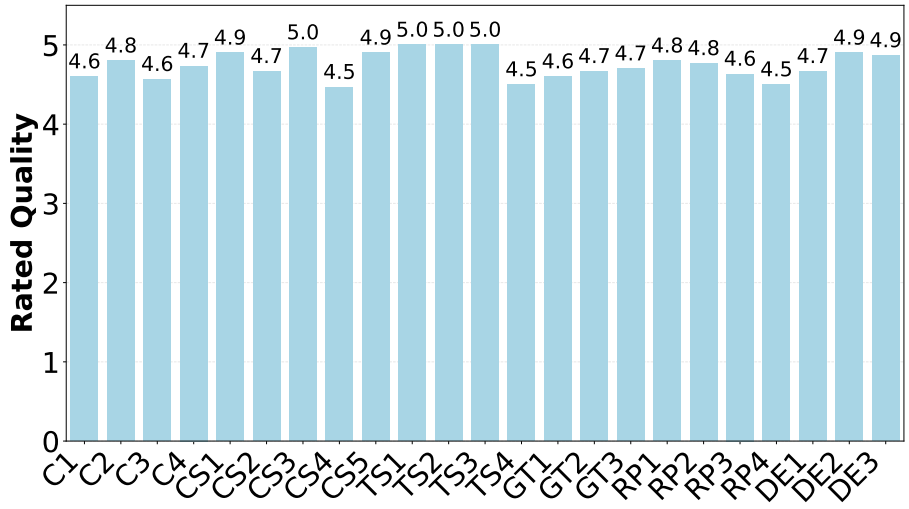


Figure 11. Human-rated quality on `MechEvalAgent` evaluations (1-5 Likert scale) on each metrics averaged across three types of tasks.

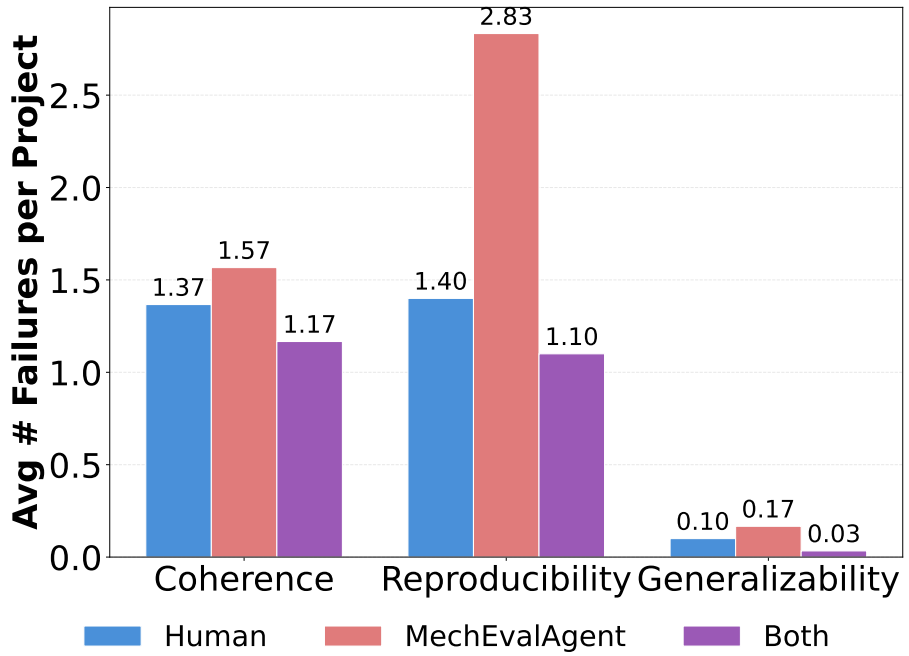
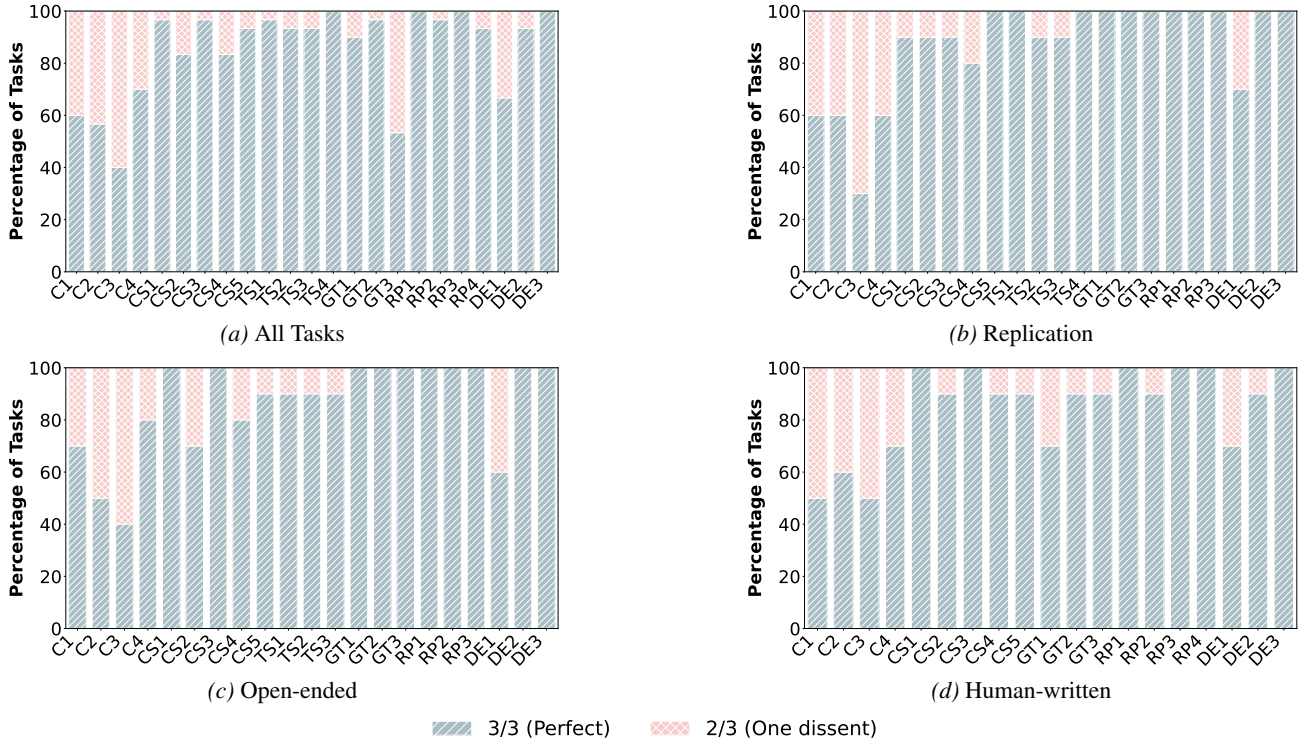
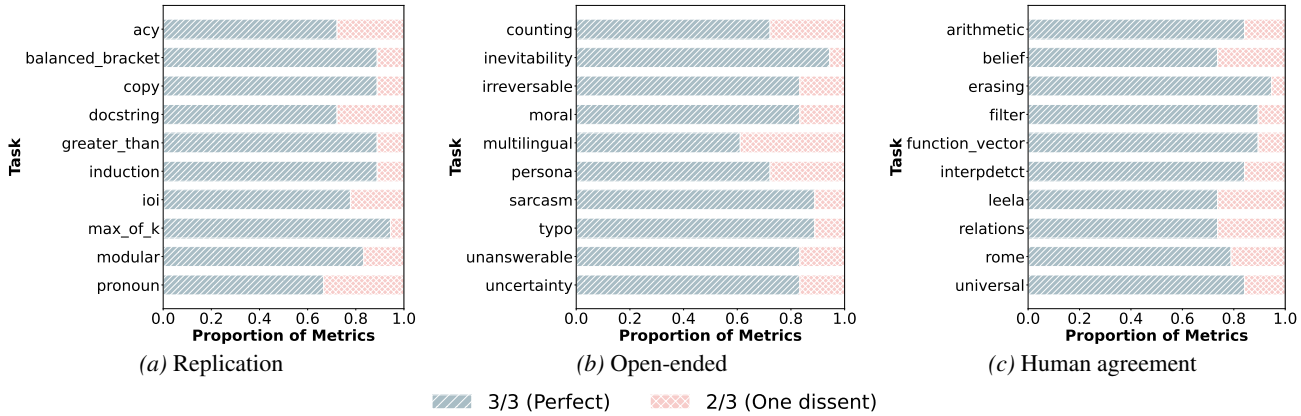


Figure 12. Average number of unique issues identified by `MechEvalAgent` per project, broken down by evaluation dimension. Reproducibility shows the most unique issues (2.83 per project), followed by coherence (1.57) and generalizability (0.17).



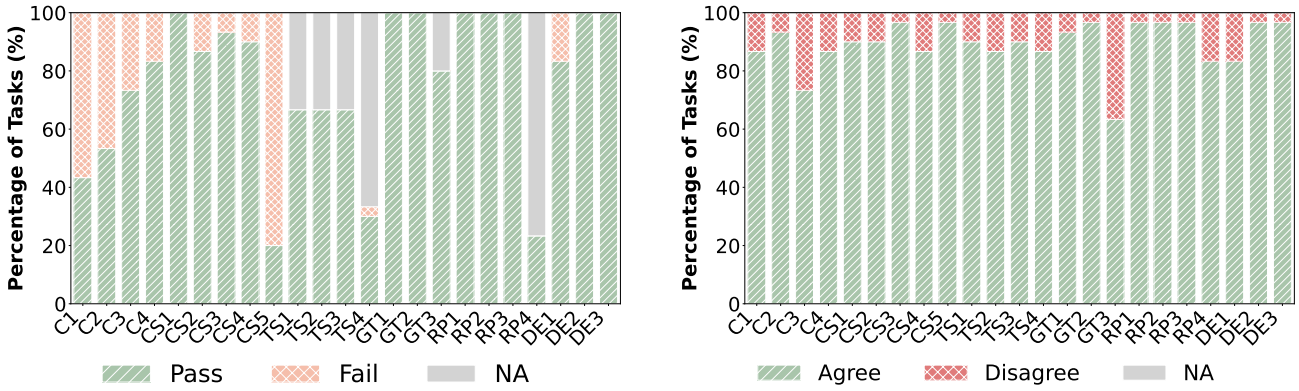
(a) All Tasks *(b)* Replication *(c)* Open-ended *(d)* Human-written

Figure 13. Evaluation stability across three runs. Bars show 3/3 (perfect) vs 2/3 (one dissent) agreement. (a) All tasks combined. (b) Replication tasks. (c) Open-ended tasks. (d) Human-written repositories. Most metrics achieve over 90% perfect agreement.



(a) Replication *(b)* Open-ended *(c)* Human agreement

Figure 14. Evaluation stability by repository type. Bars show 3/3 (perfect) vs 2/3 (one dissent) agreement across three runs. (a) Replication tasks. (b) Open-ended tasks. (c) Human-written repositories.



(a) Pass/Fail rates with majority voting

(b) Agreement with human judges

Figure 15. Evaluation results using majority voting. (a) Pass/Fail rates per metric when using majority vote (task fails if at least 2 of 3 runs fail). (b) Agreement between majority vote results and human evaluations, achieving 89.4% overall agreement.