nature portfolio

Peer Review File

Reimagining research papers as interactive and reliable AI agents

Corresponding Author: Dr James Zou

Any redactions in this file are there to maintain patient confidentiality, the confidentiality of unpublished data, or to remove third-party material.

This file contains all reviewer reports in order by version, followed by all author rebuttals in order by version.

Version 0:

Reviewer comments:

Referee #1

(Remarks to the Author)
REVIEW OF PAPER2AGENT MANUSCRIPT

SUMMARY

This manuscript by Zou and colleagues introduces Paper2Agent, an automated framework that converts research papers with associated code repositories into interactive AI agents via Model Context Protocol (MCP) servers. Paper2Agent uses a multi-agent conversion pipeline (built on Claude Code) comprising specialized sub-agents for environment configuration, tutorial extraction, tool implementation, and iterative testing to automatically generate MCP servers that package a paper's functionality as validated tools, resources (manuscript text and data links), and prompts (workflow templates); end users then query these MCP servers through a single agent (Claude Code) with natural language rather than interacting with multiple agents themselves. The authors demonstrate the approach on three computational biology papers (AlphaGenome for genomic variant interpretation, TISSUE for spatial transcriptomics, Scanpy for single-cell analysis), reporting 100% accuracy on tutorial-based queries (15/15) and novel queries (15/15) for AlphaGenome compared to 60–80% for Claude Code with repository access and 40–60% for Biomni, along with 1.8–4.6× median runtime improvements. The authors further showcase an "AI co-scientist" that combines AlphaGenome and ADHD GWAS data MCPs to prioritize rs1626703 as a likely causal variant (among 209 candidates) affecting MPHOSPH9 splicing in glutamatergic neurons based on AlphaGenome quantile scores, though this variant was already present in the original fine-mapping credible set and the computational prediction remains experimentally unvalidated. Overall, the work demonstrates impressive engineering but would be strengthened by rigorous validation of MCP correctness beyond tutorial reproduction, ablation studies isolating the contribution of MCP format versus simpler alternatives (e.g., well-structured Markdown skill files), systematic characterization of what fraction of papers can be successfully converted, demonstration with non-Claude agents to establish generalizability, and tempering of discovery claims to reflect computational hypothesis generation rather than validated findings. Overall, there is also a sense that the paper lacks quantitative assessments(most figures are illustrative examples).

STRENGTHS:

- The framework addresses genuine friction between publication and practical application of computational methods, thereby lowering technical barriers for experimental biologists and enabling faster method adoption across the research community.

- The automated conversion pipeline systematically decomposes MCP creation into specialized sub-tasks (environment setup, tutorial scanning, tool extraction, iterative testing), so that each component can focus on a well-defined objective with clear success criteria.

- The AlphaGenome agent achieves 100% accuracy on both tutorial-based (15/15) and novel (15/15) queries, representing a substantial improvement over Claude Code with repository access (60% tutorial, 80% novel) and Biomni (40% tutorial, 60% novel), thereby demonstrating measurably better reliability for end users.

- The system automatically generates 22 AlphaGenome MCP tools in approximately 3 hours without human intervention, covering single- and batch-variant scoring, tissue ontology exploration, and visualization suites, which makes the one-time upfront cost potentially worthwhile if amortized over many downstream queries.

- MCP servers are deployed remotely on Hugging Face Spaces and expose tools, resources, and prompts through a standardized protocol, in turn enabling any compatible LLM or agent to invoke functionality without custom integration code.

- The Scanpy case study demonstrates that MCP prompts can encode standardized multi-step workflows (quality control, normalization, dimensionality reduction, clustering) extracted directly from tutorials, which relieves users from needing to specify complex analysis sequences manually.

- The ADHD GWAS application shows that multiple paper MCPs (method + data) can be connected to an AI agent to generate hypotheses autonomously, thereby accelerating the process of combining new methods with new datasets from weeks of manual work to hours of automated analysis.

- The manuscript includes computational efficiency analysis showing median runtime reductions of 1.8–3.2× on tutorial benchmarks and 3.2–4.6× on novel benchmarks compared to other agents, which indicates measurable gains in user experience once the MCP is created.

- Each MCP tool embeds a traceable link to the original GitHub source code, thereby providing transparency and enabling users to verify implementation correctness or adapt code for specialized use cases.

- The paper articulates a compelling ecosystem vision where communities of paper agents could interact across disciplinary boundaries, thereby enabling AI-driven collaboration that links methods to datasets or combines insights from disparate domains.

WEAKNESSES & REQUESTS

- The manuscript does not report what fraction of computational papers have sufficiently well-maintained codebases to enable successful agentification, nor does it provide systematic evaluation across a representative sample of papers beyond three exemplar cases from computational biology, so the practical scope and generalizability of the approach remain unclear; include a survey of 50–100 randomly selected computational papers assessing conversion success rates and characterizing features that predict success or failure.

- All three case studies (AlphaGenome, TISSUE, Scanpy) represent exceptionally well-documented projects (including one fron the authors themselves) with active maintenance and clear tutorials, so they may not reflect the typical state of research code; evaluate Paper2Agent on papers with moderate documentation quality, older codebases, or partial implementations to characterize robustness.

- The benchmarking sample size is limited to 15 tutorial-based and 15 novel queries per agent, which provides insufficient statistical power to claim 100% accuracy as generalizable performance; expand to at least 100 diverse queries per agent including intentionally challenging edge cases, and report confidence intervals and statistical significance tests.

- The "novel" queries for AlphaGenome appear to be structural variations of tutorial queries (e.g., different chromosomal positions, tissues, or modalities) rather than conceptually new analysis types, so they may not adequately test generalization beyond parameter variation; include queries requiring multi-step reasoning, tool composition, or approaches not directly demonstrated in tutorials.

- The paper claims tools are "validated" through iterative testing that verifies reproduction of tutorial outputs, but this constitutes training and testing on the same distribution, thereby creating a risk of overfitting where tools reproduce expected outputs for the wrong reasons; implement train/test splits using different tutorials for validation than for tool creation, and include adversarial examples designed to expose shallow pattern matching.

- The manuscript states that tools which "repeatedly fail" are excluded from the MCP but does not report how many tools were attempted versus successfully retained, so the 100% accuracy claim may reflect survivorship bias; report the fraction of attempted tools that pass validation for each case study and provide failure case analysis characterizing why excluded tools could not be salvaged.

- The comparison with Claude Code + repository access does not account for the ~3 hours of upfront MCP creation time, so the reported 1.8–4.6× speedup is misleading without amortization analysis; report the number of queries needed before Paper2Agent becomes cost-effective compared to direct repository use, including API costs for both Claude and AlphaGenome.

- The paper positions MCP as central but does not compare with simpler alternatives such as well-structured Markdown skill files (broadly applicable but formalized recently with Claude's new Skills feature), so the necessity and added value of the MCP architecture remain undemonstrated; the authors should include a controlled comparison where AlphaGenome functionality is exposed via a Markdown skill file versus an MCP server, testing whether Claude Code achieves similar performance with the simpler approach.

- The entire implementation relies on Claude Code (claude-sonnet-4-20250514) and the multi-agent orchestration is built specifically for this platform, so generalizability to other AI systems is unknown and the framework may represent a Claude-specific application rather than a reusable contribution; test Paper2Agent with at least one non-Claude agent (e.g., Codex CLI or even an open model via OpenCode eg Qwen3-coder) and report whether comparable conversion quality can be achieved.

- The comparison with Claude Code + repository is confounded because the baseline ("Claude + Repo") appears to receive only the code repository and query, while Paper2Agent receives the repository plus the paper manuscript, structured tool definitions, and pre-validated workflows, so it remains unclear whether performance gains stem from the MCP infrastructure or simply from providing the manuscript text as context; include a comparison where Claude Code receives both the paper PDF and repository (matching Paper2Agent's information access) to establish whether the MCP format and automated conversion provide measurable benefit beyond giving Claude Code equivalent information directly.

- The manuscript does not evaluate the contribution of individual sub-agents in the multi-agent conversion pipeline (environment-manager, tutorial-scanner, tool-extractor-implementor, test-verifier-improver), so it remains unclear which components are essential versus dispensable; conduct ablation studies removing or simplifying each sub-agent to assess whether the full multi-agent architecture is necessary or whether simpler approaches (e.g., a monolithic conversion script or manual curation of specific steps) could achieve comparable MCP quality with less complexity.

- The ADHD case study claims the AI co-scientist "identified new splicing variant associated with ADHD risk," but rs1626703 was already in the fine-mapping credible set from the original GWAS paper and the AlphaGenome predictions are computational rather than experimentally validated, so this represents hypothesis prioritization rather than discovery; revise all "discovery" and "identified" language to "prioritized" or "generated computational hypothesis," especially in the abstract which is particularly misleading for a high-profile venue.

- The AlphaGenome agent prioritizes SORT1 (quantile score 0.99982) over the original paper's CELSR2 and PSRC1 (both 0.99998), but these quantile scores are functionally indistinguishable (all in top 0.02% of variants, differing by only 0.016 percentile points) and all three genes show significant eQTL associations in GTEx liver tissue, thereby illustrating that AlphaGenome provides no discriminatory power at complex GWAS loci where multiple nearby genes have extreme predictions; frame this result as "rapid generation of alternative hypotheses for human evaluation" rather than improved inference, since the agent essentially applied functional reasoning to break ties between computationally equivalent predictions.

- The manuscript does not report inter-rater reliability for ground truth validation of novel queries, which relies on "manual execution of the original AlphaGenome code" by human experts, so the objectivity and reproducibility of the 100% accuracy claim remain uncertain; include multiple independent expert raters and report inter-rater agreement statistics.

- The paper does not characterize agent failure modes such as handling of ambiguous queries, requests outside the paper's scope, or malformed inputs, so users cannot distinguish agent limitations from paper limitations and false positive rates are unknown; systematically evaluate a set of out-of-scope or adversarial queries and report how often the agent incorrectly claims capability versus appropriately declines.

- All case studies are from computational biology and bioinformatics, so generalizability to other scientific domains (chemistry, physics, machine learning, social sciences, wet-lab experimental papers) is unsupported; I would suggest demonstrating Paper2Agent on at least two papers from non-biology fields or explicitly scope the contribution to computational biology with empirical justification for this limitation.

- The manuscript defers critical algorithmic details to the GitHub repository, including how tutorials are selected from repositories, how tools are extracted from tutorial code, what constitutes "repeated failure" for tool exclusion, and how MCP prompts are constructed, thereby making it difficult to evaluate reproducibility of Paper2Agent itself; perhaps move sufficient detail into Extended Methods with pseudocode for key procedures so that the paper is self-contained enough for independent reimplementation.

- Statistical rigor is lacking, notably in Figure 2C which compares computational efficiency across agents but reports only median speedup without confidence intervals, significance tests, or even individual data points despite small sample sizes (15 queries per benchmark), so the reliability of claimed performance improvements remains uncertain; apply statistical tests (e.g., Wilcoxon signed-rank for paired runtime comparisons) and report confidence intervals to establish whether observed differences are statistically significant, and note that the predominance of qualitative schematic figures over quantitative analysis figures throughout the manuscript is unusual for a methods paper and limits rigorous evaluation of performance claims.

- The paper does not discuss maintenance burden as papers and codebases evolve (e.g., when AlphaGenome API changes or dependencies break), so the sustainability and economic model for agent ecosystems remain unaddressed; I suggest providing a concrete maintenance plan and discussing who bears responsibility for keeping agents synchronized with upstream changes.

- The manuscript does not provide a cost analysis including Claude API fees, domain-specific API costs (e.g.,

AlphaGenome), compute resources for MCP creation, and deployment hosting fees, so the practical feasibility for typical research groups is uncertain; report total costs for creating each case study MCP and estimate ongoing costs for query execution and maintenance.

- The Scanpy MCP prompts encode a "standard" preprocessing and clustering pipeline that may not suit all single-cell datasets, and while the prompt instructs the agent to "inspect the data first," there is no validation that appropriate parameter adjustments actually occur, thereby creating a risk of cargo-cult analysis where steps are followed without domain-appropriate customization; please include examples demonstrating that the Scanpy agent adapts pipeline parameters based on dataset characteristics (e.g., mitochondrial percentage thresholds, resolution parameters).

- The paper briefly mentions "security defaults" for MCP servers but provides no threat model, vulnerability analysis, or discussion of what happens if malicious code in a paper's repository gets exposed through an MCP tool, so security implications remain unaddressed; maybe discuss intellectual property considerations when third parties create agents from published papers, and clarify attribution chains if agent-generated hypotheses lead to downstream discoveries.

- The paper does not report whether Paper2Agent inherits bugs from the original codebase, so if a paper's implementation contains errors, these may be propagated and potentially amplified in the generated MCP tools; maybe discuss how Paper2Agent handles codebases with known issues and whether the validation process can detect inherited bugs versus only confirming consistency with potentially flawed tutorial outputs.

(Remarks on code availability)

Referee #2

(Remarks to the Author)
A. Summary of the key results

The article introduces Paper2Agent, an automated multi-agent pipeline that converts a research paper + public codebase into a runnable Model Context Protocol (MCP) server paired with a conversational agent. The system implements several components--including the environment-manager, tutorial-scanner, tutorial-tool-extractor-implementor, and test-verifier-improver--that produce MCP tools, resources, and prompts.

The authors evaluate Paper2Agent on three biological case studies: AlphaGenome (genomics), TISSUE (spatial transcriptomics), and Scanpy (single-cell analysis). For AlphaGenome, Paper2Agent automatically generated 22 MCP tools in ~3 hours and achieved 100% accuracy on 15 tutorial-based and 15 novel benchmark queries (manually graded vs. ground-truth code execution), while also reporting run-time speedups relative to Biomni and Claude + Code repo. Finally, the authors illustrate cross-agent integration (AlphaGenome + ADHD GWAS), where an AI co-scientist prioritized causal variants across 39 risk loci and proposed mechanistic hypotheses.

B. Originality and significance

The manuscript presents a novel system-level contribution: automated conversion of research papers into interactive MCP servers + conversational agents. While the broader idea of executable/reproducible research—e.g., executable papers, Jupyter notebooks, Papers With Code—has a long history, Paper2Agent's emphasis on automated tool extraction, structured MCP interfaces, and agent integration represents a meaningful extension of this paradigm.

In terms of technical novelty, the approach appears to build upon and combine several prior ideas, but remains incremental relative to existing platforms that expose runnable artifacts—such as CodeOcean (https://tech.cornell.edu/built/code-ocean/) and Binder (https://mybinder.org/). The manuscript would benefit from citing these tools explicitly and clarifying how Paper2Agent differs from them.

In terms of significance, Paper2Agent has potentially high practical impact: reducing barriers to method adoption and enabling agentic workflows is highly valuable for computational biology and other computational science more broadly.

C. Data & methodology: validity of approach, quality of data, quality of presentation

The workflow used to convert a codebase into an MCP server (codebase extraction, environment configuration, tool synthesis, testing, and refinement) is reasonable. Using MCP as a standard interface for LLM integration is a sound design choice. The manuscript is generally well written, and the provided code repository enables reproducibility. For improved readability, the authors should include a clearer description of how the MCP servers are created end-to-end.

D. Appropriate use of statistics and treatment of uncertainties

In the AlphaGenome benchmarking experiment, accuracy is reported without uncertainty estimates. Given that LLMs are stochastic, it would be useful to run the AlphaGenome Agent, Claude + repo, and Biomni multiple times to report variance or confidence intervals around performance.

E. Conclusions: robustness, validity, reliability

The three biological case studies demonstrate that Paper2Agent can effectively automate runnable agents for well-documented research papers. While this is promising, the evidence for general robustness is limited — the evaluated papers have actively maintained codebases and are likely easier than the average research paper, which many lack comprehensive documentation. The authors do acknowledge these limitations in cases where code is incomplete or poorly documented.

F. Suggested improvements: experiments, data for possible revision

1. Adaptation and training: In many cases, methods require tuning for specific use cases (e.g., hyperparameter adjustment). Can Paper2Agent handle such adaptation? Additionally, is it capable of training models, or is it limited to running pre-existing code?

2. Ablation studies: The workflow includes multiple sub-agents performing distinct tasks. An ablation study—e.g., disabling the test-verifier or tutorial-scanning components—would clarify the contribution of each component to overall performance.

3. Failure modes and robustness: Quantify failure rates, such as when MCP fails to generate tools or when tests fail and are pruned. should be quantified.

4. Hallucination: Has the hallucination rate been evaluated? One approach is to introduce adversarial inputs, such as modified READMEs or misleading tutorials.

5. Performance on incomplete or poorly documented repositories: The authors acknowledge that a limitation of Paper2Agent arises when the code repository is not well documented. The examples in the manuscript all use polished, well-maintained repositories, but in practice, many papers have incomplete or poorly maintained code. Therefore, the authors should evaluate Paper2Agent on such repositories to better reflect real-world scenarios. One approach could be to intentionally hide parts of the repositories used in the article and observe the agent's response. An even stronger demonstration would be to include case studies that more closely reflect real-world settings.

6. Statistical evaluation: Since LLMs are stochastic, running AlphaGenome Agent, Claude + repo, and Biomni multiple times to report variance or confidence intervals would strengthen the analysis.

7. Readability and clarity: The manuscript should explicitly list the contributions of Paper2Agent. In the extended methods, the sub-agents created for their workflow are described, but their creation process and corresponding inputs/outputs are unclear. Details on constructing the MCP servers are also missing and should be included.

8. Robustness evaluation: It will be important to include more challenging, real-world case studies — e.g., scenarios with missing code and incomplete documentation.

G. References: appropriate credit to previous work?

Overall, the paper cites relevant prior work, and the experimental comparisons against Claude + Code and Biomni are appropriate. As noted earlier, the authors should clarify how Paper2Agent differs from existing platforms that convert code repositories into executable artifactors, and include those systems explicitly in the related work discussion.

H. Clarity and context: lucidity of abstract/summary, appropriateness of abstract, introduction and conclusions

The problem statement is clear and well motivated. The proposed solution is intuitive and presented in a reasonably accessible manner, and the use cases appear practically meaningful. The conclusion highlights the strengths of Paper2Agent and also acknowledge the limitations and future directions.

Minor comments

Figure 1B suggests that the codebase is identified from the paper, whereas the extended methods section states that the codebase is provided as part of the input. Please clarify what is expected as input to Paper2Agent.

The subplots in Figure 4C are difficult to read, particularly the cell type annotations. Increasing the font size or improving contrast would make the figure more interpretable.

(Remarks on code availability)
The code is well-documented and includes multiple READMEs to support replication of the analyses in the manuscript. The

instructions cover both basic and advanced usage scenarios (e.g., target tutorial processing and repositories which need authentication). In addition, the MCP servers for the three showcased code repositories have been publicly released on Hugging Face, further increasing usability.

Referee #3

(Remarks to the Author)
The manuscript introduces Paper2Agent, a systems framework that converts a research paper together with its public codebase into an MCP-based AI agent. The framework constructs an MCP server exposing executable tools, static resources, and workflow prompts derived from the paper and its tutorials, and connects this server to an LLM-based agent for natural-language interaction. The approach is demonstrated on three computational biology codebases with an additional illustrative application combining AlphaGenome with ADHD GWAS fine-mapping data.

The paper presents a clear and coherent systems abstraction for agentifying computational methods via MCP. The engineering pipeline is carefully designed, and the chosen case studies are relevant to computational biology and demonstrate that for well-maintained repositories, the framework can reproduce tutorial outputs and enable interactive use. This is a thoughtful and practically motivated systems paper that addresses a real pain point in method reuse and reproducibility. However, the empirical evaluation is narrow, with quantitative benchmarking limited to a single method (AlphaGenome) and a small, curated set of queries. As a result, claims regarding reliability, reproducibility, and generality are not fully supported. In addition, statistical quantification and analysis is minimal, and robustness to prompt variation, model choice, or run-to-run stochasticity is not evaluated.

Major points
1. The contribution is mainly a systems and engineering integration, rather than a new algorithmic or modeling advance. The idea of enabling interactive execution of scientific methods builds on existing directions including executable papers, paper-to-code systems, and scientific agents. The main added value here is the combination of MCP-based packaging, automated tool extraction from tutorials, and test-verified execution, which is a great advance. But it would be important to clarify the novelty relative to existing work (e.g., executable papers, Paper2Code/PaperCoder, Biomni, CellVoyager). A comparison table identifying differences (e.g., tool synthesis vs. repo generation, multi-paper composition, etc) could help clarify what Paper2Agent enables that prior systems do not.

2. The multi-agent pipeline (environment setup, tutorial scanning, tool extraction, testing and refinement) is reasonable. However, the evaluation conflates feasibility demonstrations with claims of general reliability. Only AlphaGenome is evaluated quantitatively, while Scanpy and TISSUE are assessed qualitatively via consistency with human-executed workflows. The ADHD GWAS example is best interpreted as an illustrative demonstration rather than a benchmark. The authors could add a small but systematic benchmark across a more diverse set of repositories (e.g., Python-only, R-based, mixed-language, limited documentation), reporting success/failure rates and failure modes.

3. Accuracy is reported as a single point estimate (on AlphaGenome queries) based on manual grading. There are no confidence intervals, no assessment of inter-rater agreement, and no robustness analysis with respect to prompt phrasing or repeated runs. The manuscript acknowledges these limitations, but the conclusions lean on strong reliability language. At minimum, the authors should report confidence intervals for success rates, run-to-run variance, and evaluate sensitivity to paraphrased queries.
4. Paper2Agent depends on the quality and completeness of the underlying codebase, and that not all papers can be faithfully converted into agents. If this is the case and the agent is not useful for papers without complete codebases and documented tutorials, then the added advantage may be minimal.

Minor points
1. The definition of “novel queries” should be clarified operationally
2. A rubric for manual grading would improve transparency.
3. The Scanpy and TISSUE examples would benefit from even small quantitative evaluations
4. Clarify what is a passing test (Fig. 1B) and what is the policy when failures persist
5. What are hardware specs for runtime test (Fig. 2C)
6. Fig 2D: Include summary statistic: e.g., number of loci/tasks where the full report generation completed without manual intervention, and common failure modes.
7. Fig 3C: how do you quantify matching? Report quantitative similarity and benchmarks. Are these results reproducible?

(Remarks on code availability)
The repository provides a README with a clear high-level description, example commands, and a shell-based entry point that documents how to run the pipeline on a target GitHub repository, including options for tutorial filtering and benchmarking, which is helpful for technically experienced users.

Referee #4

(Remarks to the Author)
I co-reviewed this manuscript with one of the reviewers who provided the listed reports.

(Remarks on code availability)

Referee #5

(Remarks to the Author)
I co-reviewed this manuscript with one of the reviewers who provided the listed reports.

(Remarks on code availability)

Version 1:

Reviewer comments:

Referee #1

(Remarks to the Author)
The authors have performed a substantial revision, adding a 100-paper evaluation, ablation studies, cross-platform testing, and a new discovery case study. I acknowledge the effort and recognize that many specific requests from the first round have been addressed with new data. However, the field of AI agents has moved rapidly since the original submission, and I have significant concerns about the durability and significance of the contribution.

Improvements:

The large-scale evaluation (74/100 papers successfully agentified, 91.2% accuracy on 300 benchmark questions) with detailed failure mode characterization provides the systematic scope assessment that was previously missing. The ablation studies convincingly justify specific architectural choices — the monolithic agent's complete failure (0/3 runs), the accuracy drop without test-verification (from ~99% to ~69%), and the MCP vs. skill file gap (99.3% vs 73.3%) demonstrate that each design choice contributes meaningfully. The OpenCode backend evaluation (98.7% tutorial, 97.3% novel accuracy) establishes that Paper2Agent is not inherently Claude-specific, addressing a major generalizability concern from the first round. The GPR137/psoriasis case study adds a validation component using existing scCRISPRi and Perturb-seq data, though as noted below this remains a computational analysis rather than independent experimental validation. Statistical rigor is now appropriate throughout, with bootstrap SEs, confidence intervals, and paired significance tests across substantially expanded sample sizes.

Remaining concerns:

1. General-purpose coding agents are increasingly capable of doing on-the-fly what Paper2Agent pre-computes. The paper's own data illustrates this: Claude Code with direct repository access already achieves 80.3% accuracy on the large-scale benchmark, compared to Paper2Agent's 91.2%. This is an 11 percentage point gap, real but modest. The baseline was measured using claude-sonnet-4-20250514, a model now several generations behind current frontier capabilities. The gap has almost certainly narrowed since the experiments were run and will likely continue to shrink as models improve.

The economics reinforce this concern. At ~$15 and 2.4 hours per paper on average, with cost break-even at ~245 queries, most niche computational tools will never reach the threshold where pre-building an MCP server pays off. The speed advantage (1.6 vs 4.3 min/query) is real but increasingly marginal as inference costs drop. The authors frame agentification as a "one-time community investment," but this requires adoption infrastructure (journals or communities maintaining agents) that does not yet exist and whose feasibility is undemonstrated.

More fundamentally, a pre-computed MCP server is a frozen artifact that decays as upstream codebases evolve (API changes, dependency updates, data format shifts). The authors acknowledge this and propose maintenance agents (Claude Code Web/Jules) for periodic re-validation — but this concedes that the MCP server requires ongoing energy to prevent decay. A general-purpose agent interacting with the live repository requires no maintenance and always operates on the current codebase. The maintenance burden may ultimately undermine the efficiency gains that justify the approach.

2. Related to the above, the new large-scale evaluation data suggests the contribution may be closer to optimized retrieval-augmented generation (RAG) than a paradigm shift. The structured resource layer (for non-computational papers) achieves 89.0% vs 82.0% for a Claude browser-use baseline, a 7 percentage point gap. The executable tool layer adds more value (91.2% vs 80.3%), but the multi-agent conversion pipeline is fundamentally a wrapper around the underlying LLM's coding ability. It does not introduce new algorithms, new scientific reasoning, or new domain knowledge; it reorganizes existing code into a format that is easier for an LLM to invoke. As coding agents improve at reading documentation and executing code on first attempt, this reorganization becomes less necessary. The presented data does not suggest that Paper2Agent adds any major independent capability beyond what the base model provides, and the value of the wrapper will erode as the base models catch up.

3. The authors clarify that Paper2Agent does not access the manuscript during evaluation, only the pre-built MCP tools and metadata. The "Claude + Repo" baseline receives both the full code repository and the paper. This means the baseline has equivalent raw information access, yet Paper2Agent still outperforms it by 11 points. The gain therefore stems from the pre-

computed tool extraction and validation : the codebase has been distilled into structured, pre-tested API calls. While this is a legitimate engineering advantage, it is the kind of advantage that general-purpose agents will increasingly replicate on the fly as they become better at reading documentation and executing code correctly on first attempt. This reviewer's own experience is that current frontier models (e.g., Claude Opus 4.6) can already read a paper's repository, understand its API, and execute correct tool calls without pre-computed scaffolding, suggesting the baseline used in this study substantially underestimates current general-purpose agent capability.

4. The authors inspected MCP implementations for hardcoded artifacts and added an automated reviewer agent. However, the fundamental design (tools are built and validated against tutorials, then evaluated on queries derived from those same tutorials) means the training and evaluation distributions remain coupled. The 100% accuracy on "novel" AlphaGenome queries remains statistically suspicious; in real-world scientific use, "novel" means unforeseen edge cases, and perfect accuracy on 15 queries suggests the benchmark is still too close to the training distribution. The open-ended queries (84% accuracy on 15 queries for a single case study) provide the strongest evidence against pure pattern matching but are too limited to establish robust generalization.

5. In response to my original request for evidence that the Scanpy agent adapts pipeline parameters based on dataset characteristics, the revised manuscript claims that "the agent adaptively adjusts parameters based on data characteristics: selecting organism-appropriate mitochondrial gene prefixes, reducing HVG counts for targeted gene panels, and applying tissue-specific marker genes." This is asserted in a single sentence without systematic evidence. A supplementary table showing specific parameter choices across diverse datasets would be needed; as written, it could equally reflect the agent following a rigid template that happens to work on the tested datasets.

6. The psoriasis/GPR137 case study (Figure 6) is presented as experimentally validated, but the validation is a secondary analysis of existing experimental datasets (scCRISPRi and Perturb-seq), not new experiments. No new wet-lab validation was performed. Two additional points deserve more transparent treatment: (1) the human researcher selected the validation strategy (signature correlation analysis) from among 10 candidates proposed by the AI co-scientist; this human-in-the-loop step is essential but underemphasized, risking overestimation of autonomous discovery capability; (2) the Spearman correlation for GPR137 is significant only under stimulated conditions (Stim8hr: rho = 0.61, p = 3.8e-3; Stim48hr: rho = 0.63, p = 4.7e-3) but not at rest (rho = 0.29, p = 0.21). The AI co-scientist did not predict this condition-dependence, and the paper does not discuss its biological implications. Whether the causal mechanism is activation-dependent rather than constitutive matters for the interpretation of GPR137's role in psoriasis.

For the ADHD case, while "identified" has been changed to "prioritized," the framing throughout (including Figure 5 caption: "revealing a plausible causal mechanism for ADHD risk") still reads as stronger than what the data support.

7. The 74% success rate means 1 in 4 papers cannot be agentified at all. Among those that succeed, ~9% of queries produce incorrect answers. For scientific applications where correctness matters, this combined failure profile deserves more prominent discussion.

8. The adversarial self-repair tests (36/36) cover a limited set of failure modes (path errors, missing dependencies, typos, deprecated APIs). Real-world repository degradation (breaking changes in upstream libraries, deprecated web APIs, data format evolution) was not tested.

9. The non-biology evaluation (5 papers, 17 questions) remains thin for supporting a cross-domain generalization claim.

Overall assessment:

The revised manuscript presents well-engineered work and the authors have been responsive to reviewer feedback with substantial new analyses and results. However, the central value proposition, that pre-computing MCP tools provides substantial benefit over giving a capable coding agent the repository directly, rests on a modest accuracy gap measured against an already-outdated baseline. The field of AI agents is moving extremely fast; what may have required a complex multi-agent pipeline in mid-2025 may be achievable by a general-purpose agent (like Opus 4.6 in Claude Code) in 2026. My assessment is that at this stage, the contribution is primarily one of engineering optimization (2.8x cheaper and 6.3x faster per query, 11 points more accurate) rather than a fundamental advance in how scientific knowledge is disseminated or used. I would recommend this work for a high-impact methods venue, where the technical contributions would be appropriately valued, rather than Nature where the expectation is for broader conceptual advances.

(Remarks on code availability)

Referee #2

(Remarks to the Author)
The authors have provided a detailed and thoughtful response to the comments, and the updated manuscript is substantially strengthened by the addition of further empirical results and improved clarity. In particular, the expanded experimental evaluation demonstrating the generalizability of Paper2Agent across diverse types of papers, the analysis of failure modes, the inclusion of uncertainty measures, quantification of hallucination rates, and systematic ablations of system components all meaningfully enhance the rigor and impact of the work. The comparisons with existing executable artifact platforms such

as CodeOcean are also valuable in this rapidly evolving space. The added statistical analyses appear appropriate, and the reported p-values support the stated conclusions. Overall, the major concerns raised in the initial review have been satisfactorily addressed.

A few points remain that could further improve the manuscript:

1. CodeOcean has recently introduced an AI agent platform with an MCP server (https://codeocean.com/blog/trusted-agents). Paper2Agent couldbe compared against this system.

2. Although Paper2Agent outperforms previous baselines, its accuracy on benchmark questions is not perfect, implying that incorrect answers may occasionally be produced. This raises an important issue: how should users assess and trust responses to novel scientific questions, particularly in biological domains where errors could lead to incorrect conclusions? The authors should more explicitly caution readers about this limitation and clarify appropriate usage scenarios.

3. The manuscript should clarify whether executable tutorials are required for the MCP tool to function. Additionally, can Paper2Agent automatically generate its own tutorials based on repository code?

4. For the adversarial stress tests, it would strengthen the analysis to include the performance of functional MCP servers on the case studies and benchmark tasks.

(Remarks on code availability)

Referee #4

(Remarks to the Author)
I co-reviewed this manuscript with one of the reviewers who provided the listed reports.

(Remarks on code availability)

Referee #5

(Remarks to the Author)
I co-reviewed this manuscript with one of the reviewers who provided the listed reports.

(Remarks on code availability)

Version 2:

Reviewer comments:

Referee #1

(Remarks to the Author)
This revision adds updated Claude Opus 4.6 and Sonnet 4.6 baselines, expanded open-ended and non-biology benchmarks, and repository-drift stress tests. Some of my requests are handled well: the new Scanpy table convincingly shows the agent adapting markers to tissue and disease context. The GPR137 signal's restriction to stimulated cells is now reported and explained. The ADHD language is appropriately softened.

Unfortunately, one of my main concerns from last round is only partly resolved. I had argued that the accuracy gap reflected an outdated baseline that newer models would erase. On AlphaGenome they tested this directly, and an Opus 4.6 agent with repository access still trailed Paper2Agent. However, that test covered AlphaGenome alone; the large-scale benchmark that carries the central scalability claim (300 questions across the 74 agentified papers) still runs on the older Sonnet 4 model. So whether Paper2Agent's advantage persists as base models improve remains, in my view, open.

My other earlier concern also remains: the tools are built and validated on the same tutorials they are later tested on, and I would argue the new data recasts this as a question about what the benchmark scores actually measure. For the computational benchmarks, the tools are extracted from and validated against the same reference code that defines the ground truth, so these scores largely measure faithful reuse of one reference implementation. For a paper agent, that conformity is partly the point, but it means the results support reproducibility and reuse more directly than claims about scientific reasoning. An agent that writes a valid but different implementation could be deemed wrong unless equivalence is checked separately. Rather than a new experiment, I would value a subset audited for scientific equivalence, or a count of how many baseline answers were truly wrong rather than merely different.

Altogether, this leaves me unconvinced on significance. The engineering is genuinely strong: Paper2Agent converts a repository into a validated, queryable MCP server automatically, and repairs broken codebases. But the data substantiate the operational claims, eg improved reliability, reproducibility, and reuse of existing methods, more than the conceptual

claim of a new paradigm for scientific communication. Establishing the latter (in my view needed for publication at Nature) would require isolating a capability attributable to agentification itself, rather than to the base model operating on the same repository, and the current experiments do not separate the two. The economics, including break-even and who maintains long-tail agents, are also not empirically resolved. My two remaining reservations are what the computational benchmarks actually measure and whether anything carries the significance claim beyond the engineering.

(Remarks on code availability)

Referee #2

(Remarks to the Author)
The authors have provided detailed responses and further experimental results to the questions raised in the response to the revision. The additional updates to the manuscript have helped in improving the clarity, limitations, and robustness of Paper2Agent. I thank the reviewers for trying out to make the CodeOcean MCP server work. Also, it is good to mention that the outputs from Paper2Agent should not be used directly for scientific discovery without human verification since it can still make mistakes. This will help the readers understand the intended use of Paper2Agent. It is also great to see that Paper2Agent is robust to missing tutorials and adversarial executions. For the adversarial stress test response, it seems like the authors have misplaced revised manuscript text in the response. Hopefully the results from the repaired MCPs are included in the revised manuscript. Overall, the remaining questions and concerns after the initial revision have been satisfactorily addressed.

(Remarks on code availability)

Referee #5

(Remarks to the Author)
I co-reviewed this manuscript with one of the reviewers who provided the listed reports.

(Remarks on code availability)

Overall Comments to Reviewers:

We sincerely thank the reviewers for their thorough evaluation of our manuscript.

We are grateful that the reviewers recognized Paper2Agent as addressing "genuine friction between publication and practical application of computational methods" (Reviewer 1), presenting "a novel system-level contribution" with "potentially high practical impact" (Reviewer 2), and providing "a clear and coherent systems abstraction for agentifying computational methods" (Reviewer 3). We appreciate Reviewer 1's recognition that the framework "lowers technical barriers for experimental biologists" and that "the automated conversion pipeline systematically decomposes MCP creation into specialized sub-tasks." We thank Reviewer 2 for noting that "the workflow used to convert a codebase into an MCP server is reasonable" and that "the provided code repository enables reproducibility." We also thank Reviewer 3 for highlighting that "the engineering pipeline is carefully designed" and that "this is a thoughtful and practically motivated systems paper that addresses a real pain point in method reuse and reproducibility."

The reviewers' insights have been invaluable in helping us strengthen our empirical evaluation and clarify the scope and limitations of Paper2Agent. We have carefully incorporated all feedback in our updated manuscript. To this end, we have added substantial new experiments demonstrating the generality, scalability, and robustness of Paper2Agent across diverse scientific domains and repository conditions.

High-level summary of new results:

To address the reviewers' feedback regarding the generalizability, robustness, and efficiency of Paper2Agent, we have performed extensive new experiments and benchmarking. Specifically, we executed an end-to-end processing and evaluation pipeline for 131 additional papers, performed exhaustive ablation studies to validate our core architectural decisions, and integrated comprehensive statistical rigor (including P-values and confidence intervals) across all benchmarks. Furthermore, we include a new discovery case study that demonstrates the framework's practical utility in genetics. Below is a summary of the key new results:

1. **Large-scale Evaluation of generality, scalability, and efficiency**: We conducted new systematic evaluation for Paper2Agent beyond the original case studies to assess generalizability:

    - **100 computational biology papers**: We randomly selected 100 papers from bioRxiv's bioinformatics category and processed them using Paper2Agent end-to-end without manual intervention. Of these, 74% were successfully agentified into executable MCP tools. Across these papers, Paper2Agent achieved 91.2 ± 1.6% accuracy on execution-based benchmark tasks, compared with 80.3 ± 2.3% for a strong baseline using Claude Code with direct repository access ($p < 0.0001$, bootstrap test). We additionally analyzed common failure

modes among the remaining papers, which were primarily driven by missing code, unavailable data or model artifacts, or unresolved environment dependencies.

- **26 non-computational papers**: We evaluated Paper2Agent on 13 bioRxiv and 13 Nature papers focused on data interpretation rather than executable methods. On 100 synthesis-based benchmark questions spanning main text and supplementary materials, Paper2Agent achieved 89.0 ± 3.1% accuracy, outperforming a Claude browser-use baseline (82.0 ± 3.8%, $p = 0.03$), while being 34× cheaper and 15× faster.

- **5 non-biology computational papers for cross-domain generalization:** We extended evaluation to diverse scientific domains including causal inference, mechanistic interpretability, natural language processing, computer vision, and tabular machine learning (TabPFN), achieving 95.3% ± 1.9% average accuracy across 17 benchmark questions. We showed that agents generated by Paper2Agent adaptively adjust parameters based on dataset characteristics and autonomously perform model training with iterative hyperparameter tuning, demonstrating capabilities beyond executing pre-existing code.

- **15 new open-ended AlphaGenome queries for complex multi-step reasoning**: We added complex queries requiring multi-step reasoning and tool composition beyond tutorial workflows. Paper2Agent achieved 84.0 ± 2.7% accuracy, significantly outperforming Claude + repository (65.3 ± 3.9%, paired t-test $p = 0.004$) and Biomni (76.0 ± 4.5%, $p = 0.035$).

- **Cost and Time Analysis:** We added a comprehensive cost and runtime evaluation of Paper2Agent. On average, successfully agentifying a computational paper requires $14.99 in API costs and 2.4 hours of fully automated processing. At query time, Paper2Agent reduces per-query cost by 1.9x ($0.20 vs. $0.38 for the baseline) and latency by 2.7x (1.6 minutes vs. 4.3 minutes for the baseline).

2. **System Robustness and Ablation Studies**
    - **Adversarial Self-Repair:** We tested 12 adversarial repository variants (injected with path errors, missing dependencies, etc.) with three independent runs. Paper2Agent successfully diagnosed and repaired 100% (36/36) of these configurations.
    - **Architectural Justification:** We conducted systematic ablation studies on the AlphaGenome repository to evaluate the necessity of Paper2Agent's design choices. By comparing our full multi-agent pipeline against five simplified variants, we confirm that modularization and automated verification are essential for reliable agentification:
        - **Monolithic Agent:** We tested a single-agent system where all sub-tasks were executed within a single 200K-token context window. This variant

failed in 100% of runs (3/3), never producing a valid MCP server. Failures were dominated by context management collapse and long-horizon planning instability, often resulting in premature workflow termination.

- **Non-Parallel Multi-Agent:** When sub-agents were forced to run sequentially, MCP construction succeeded in 3/3 runs, but runtime increased by 2.2–2.4x (mean: 99 min vs. 43 min for full system). This confirms that parallel orchestration is critical for scalable processing but not strictly required for correctness.
- **No Test-Verifier-Improver sub-agent**: Removing the automated verification loop preserved MCP construction success (3/3 runs), but caused substantial downstream performance degradation. Tutorial-task accuracy dropped from 98.7% ± 1.3% to 69.3% ± 4.5%, and novel-task accuracy dropped from 100.0% ± 0.0% to 84.0% ± 2.7% across five independent runs.
- **Markdown Skill Files:** Replacing MCP tools with Markdown skill files preserved functional execution but substantially reduced downstream reliability. Across five independent runs, tutorial-task benchmark accuracy decreased from 98.7% ± 1.3% to 70.7% ± 4.5%, and novel-task accuracy dropped from 100.0% ± 0.0% to 76.0% ± 2.7%.
- **Alternative Scaffolding:** Replacing the Claude Code backend with OpenCode preserved both MCP construction success and downstream performance. Across five independent end-to-end conversions and downstream evaluations, tutorial-task accuracy was 98.7% ± 1.3% and novel-task accuracy was 97.3% ± 1.6%.

3. **New Biological Discovery Validation**: We added a second discovery application where the AI co-scientist integrated three paper agents (AlphaGenome, MPRA-coupled scCRISPRi, Perturb-seq) to identify and experimentally validate *GPR137* as the causal gene for a psoriasis-associated variant.

4. **Clarifications and expanded methods**: based on reviewer feedback, we:
    - Clarified test passing criteria (numerical tolerance, perceptual hash distance for figures) and failure policies (6 attempts, then exclusion)
    - Added comparison with prior work (Binder, CodeOcean, Paper2Code)
    - Discussed maintenance plans using Claude Code Web/Jules for automated re-validation
    - Addressed security, intellectual property, and attribution considerations
    - Expanded Extended Methods with detailed sub-agent descriptions and pseudocode

For clarity, we use black text to indicate original reviewer comments. We use blue to highlight revisions we made to our writing, both here and the revised version of the manuscript. We use blue to respond to reviewers' comments here.

Referees' comments:

Referee #1 (Remarks to the Author):

REVIEW OF PAPER2AGENT MANUSCRIPT

SUMMARY

This manuscript by Zou and colleagues introduces Paper2Agent, an automated framework that converts research papers with associated code repositories into interactive AI agents via Model Context Protocol (MCP) servers. Paper2Agent uses a multi-agent conversion pipeline (built on Claude Code) comprising specialized sub-agents for environment configuration, tutorial extraction, tool implementation, and iterative testing to automatically generate MCP servers that package a paper's functionality as validated tools, resources (manuscript text and data links), and prompts (workflow templates); end users then query these MCP servers through a single agent (Claude Code) with natural language rather than interacting with multiple agents themselves. The authors demonstrate the approach on three computational biology papers (AlphaGenome for genomic variant interpretation, TISSUE for spatial transcriptomics, Scanpy for single-cell analysis), reporting 100% accuracy on tutorial-based queries (15/15) and novel queries (15/15) for AlphaGenome compared to 60–80% for Claude Code with repository access and 40–60% for Biomni, along with 1.8–4.6× median runtime improvements. The authors further showcase an "AI co-scientist" that combines AlphaGenome and ADHD GWAS data MCPs to prioritize rs1626703 as a likely causal variant (among 209 candidates) affecting MPHOSPH9 splicing in glutamatergic neurons based on AlphaGenome quantile scores, though this variant was already present in the original fine-mapping credible set and the computational prediction remains experimentally unvalidated. Overall, the work demonstrates impressive engineering but would be strengthened by rigorous validation of MCP correctness beyond tutorial reproduction, ablation studies isolating the contribution of MCP format versus simpler alternatives (e.g., well-structured Markdown skill files), systematic characterization of what fraction of papers can be successfully converted, demonstration with non-Claude agents to establish generalizability, and tempering of discovery claims to reflect computational hypothesis generation rather than validated findings. Overall, there is also a sense that the paper lacks quantitative assessments(most figures are illustrative examples).

STRENGTHS:

- The framework addresses genuine friction between publication and practical application of computational methods, thereby lowering technical barriers for experimental biologists and enabling faster method adoption across the research community.

- The automated conversion pipeline systematically decomposes MCP creation into specialized sub-tasks (environment setup, tutorial scanning, tool extraction, iterative testing), so that each component can focus on a well-defined objective with clear success criteria.

- The AlphaGenome agent achieves 100% accuracy on both tutorial-based (15/15) and novel (15/15) queries, representing a substantial improvement over Claude Code with repository access (60% tutorial, 80% novel) and Biomni (40% tutorial, 60% novel), thereby demonstrating measurably better reliability for end users.

- The system automatically generates 22 AlphaGenome MCP tools in approximately 3 hours without human intervention, covering single- and batch-variant scoring, tissue ontology exploration, and visualization suites, which makes the one-time upfront cost potentially worthwhile if amortized over many downstream queries.

- MCP servers are deployed remotely on Hugging Face Spaces and expose tools, resources, and prompts through a standardized protocol, in turn enabling any compatible LLM or agent to invoke functionality without custom integration code.

- The Scanpy case study demonstrates that MCP prompts can encode standardized multi-step workflows (quality control, normalization, dimensionality reduction, clustering) extracted directly from tutorials, which relieves users from needing to specify complex analysis sequences manually.

- The ADHD GWAS application shows that multiple paper MCPs (method + data) can be connected to an AI agent to generate hypotheses autonomously, thereby accelerating the process of combining new methods with new datasets from weeks of manual work to hours of automated analysis.

- The manuscript includes computational efficiency analysis showing median runtime reductions of 1.8–3.2× on tutorial benchmarks and 3.2–4.6× on novel benchmarks compared to other agents, which indicates measurable gains in user experience once the MCP is created.

- Each MCP tool embeds a traceable link to the original GitHub source code, thereby providing transparency and enabling users to verify implementation correctness or adapt code for specialized use cases.

- The paper articulates a compelling ecosystem vision where communities of paper agents could interact across disciplinary boundaries, thereby enabling AI-driven collaboration that links methods to datasets or combines insights from disparate domains.

Response: Thank you very much for your thorough and helpful comments, we truly appreciate it! We have carefully added new experiments to incorporate all of your feedback.

WEAKNESSES & REQUESTS

- The manuscript does not report what fraction of computational papers have sufficiently well-maintained codebases to enable successful agentification, nor does it provide systematic evaluation across a representative sample of papers beyond three exemplar cases from

computational biology, so the practical scope and generalizability of the approach remain unclear; include a survey of 50–100 randomly selected computational papers assessing conversion success rates and characterizing features that predict success or failure.

Response: We thank the reviewer for this suggestion. Conceptually, Paper2Agent produces two complementary components for each paper: (i) a structured resource layer that exposes the main manuscript, supplementary tables, and metadata for retrieval and question answering, and (ii) an executable tool layer that exposes the computational methods described in the paper as callable MCP tools. For computational papers and other papers, the first component is always achievable, while the second depends on the quality and agentifiability of the associated code repository.

To quantify the second component, we conducted a systematic evaluation of 100 computational research papers randomly selected from bioRxiv's bioinformatics category. We define successful agentification in terms of execution capability: a paper was considered successfully agentified if the generated MCP server completed tool extraction, execution, and passed automated testing via the test–verifier–improver agent in an end-to-end run with no human intervention. We report the overall success rate, the fraction of tools attempted versus retained, and common failure modes, and analyze predictive factors such as documentation quality, dependency completeness, and tutorial availability.

Using this definition, 74 of 100 papers (74%) were successfully agentified with executable tools. The average cost for processing one successfully agentified paper is $14.99 and average time is 2.4 hours; papers that failed to agentify typically failed early in the pipeline (average cost $9.03, average time 1.7 hours). The other 26 papers can be turned into paper agents without executable code. We evaluated these 74 paper agents on 300 benchmark questions derived from GitHub tutorials, achieving an accuracy of 91.2 ± 1.6% (mean ± bootstrap SE), significantly outperforming the baseline (Claude Code with direct repository access) at 80.3 ± 2.3% ($p < 0.0001$, bootstrap test; 95% CI of difference: [5.8%, 15.8%]). Paper2Agent answered benchmark questions at an average cost of $0.20 per query in 1.6 minutes, compared to the baseline's $0.38 per query in 4.3 minutes, indicating a 1.9x reduction in cost and 2.7x speedup in response time.

For the 26 papers that could not be fully agentified, we identified the following failure modes:

- No executable code (9 papers, 35%): repository was incomplete (e.g., "code will be released after acceptance") or contained only data files with no code.
- Missing data/model prerequisites (8 papers, 31%): required data files, model weights, or external resources not included in repository
- Environment/dependency failures (6 papers, 23%): unresolvable package conflicts or missing system libraries
- No extractable API (3 papers, 12%): the agent determined that the repository code was not generalizable for analyzing new data (e.g., reproducibility scripts hardcoded to specific datasets, GUI-only workflows)

These failure modes strongly correlate with documentation incompleteness, lack of environment specifications, and lack of executable tutorials, which together explain the majority of execution failures. In such cases, Paper2Agent's feedback can help the authors in making their code more usable.

We note that even when the executable tool layer cannot be constructed, Paper2Agent still provides value through its structured resource layer (manuscript text, supplementary tables, and metadata), enabling retrieval-augmented question answering for all papers. To benchmark this capability, we curated 100 questions from 13 bioRxiv preprints and 13 Nature articles. Paper2Agent achieved 89.0 ± 3.1% accuracy on tasks requiring synthesis of information across main text and supplementary materials, for example: "The paper reports results using Pearson correlation; reanalyze the conclusions using Spearman correlation." This significantly outperforms a strong baseline using Claude with browser-use capabilities and the paper URL, representing a strong human-assisted LLM setup, that achieved 82.0 ± 3.8% accuracy ($p = 0.03$, bootstrap test). Moreover, Paper2Agent was 34× cheaper ($0.27 vs $9.04 per question) and 15× faster (63s vs 919s per question).

Together, these evaluations demonstrate that Paper2Agent provides practical value across a broad spectrum of computational papers. For papers with well-maintained codebases, Paper2Agent enables fully automated tool construction with high accuracy on computational tasks. For the remaining papers and indeed for all papers regardless of code quality, the structured resource layer supports accurate information synthesis from main text and supplementary materials. These results establish both the scope and limitations of automated agentification: while executable tool generation depends on repository quality, the core capability of transforming papers into queryable, task-ready agents generalizes reliably across the scientific literature.

We have added those new results in our revised paper.

Revised Results section:

"Next, we conducted a large-scale evaluation across a heterogeneous corpus of research papers. We randomly selected 100 computational research papers from bioRxiv's bioinformatics category, 26 general research papers from bioRxiv and Nature that focus on data and discovery rather than executable methods, and 5 computational papers from non-biology domains including AI and statistics. All papers were processed end-to-end by Paper2Agent without manual cleanup, code modification, or intervention."

Paper2Agent produces two complementary components for each paper: (i) a structured resource layer that exposes the manuscript text, supplementary information, and metadata for retrieval and question answering, and (ii) an executable tool layer that exposes computational methods as callable MCP tools. While the resource layer is always constructible, successful tool-layer construction depends on the quality and agentifiability of the associated code repository.

“Among the 100 computational papers, 74 of 100 (74%) were successfully agentified with executable tools. Paper2Agent proposed 599 tools in total, of which 593 passed automated validation. The average cost for processing one successfully agentified paper was $14.99 with an average runtime of 2.4 hours; papers that failed typically failed early in the pipeline (average cost $9.03, average time 1.7 hours). The generated MCP server implementations were further inspected for potential shortcut learning or tutorial-specific memorization. We did not find systematic evidence of hard-coded tutorial constants, fixed dataset paths, cached outputs, or dataset-specific heuristics.

We evaluated these 74 paper agents on 300 benchmark questions derived from GitHub tutorials. Paper2Agent achieved 91.2 ± 1.6% accuracy (mean ± bootstrap SE), significantly outperforming the baseline (Claude Code with direct repository access) at 80.3 ± 2.3% ($p < 0.0001$, bootstrap test; 95% CI of difference: [5.8%, 15.8%]). Paper2Agent answered benchmark questions at an average cost of $0.20 per query in 1.6 minutes, compared to $0.38 per query in 4.3 minutes for the baseline, corresponding to a 1.9x cost reduction and 2.7x speedup. For the remaining 26 computational papers that could not be fully agentified, dominant failure modes included missing executable code (35%), missing data or model artifacts (31%), environment or dependency failures (23%), and non-generalizable scripts (12%). These failures strongly correlate with documentation completeness, environment specification quality, and the presence of executable tutorials.

Even when executable tools cannot be constructed, Paper2Agent still provides value through its structured resource layer. Across 26 non-computational papers (13 bioRxiv and 13 Nature papers), we curated 100 synthesis-based benchmark questions spanning main text and supplementary materials. Paper2Agent achieved 89.0 ± 3.1% accuracy, significantly outperforming a strong Claude browser-use baseline (82.0 ± 3.8%, $p = 0.03$), while being 34× cheaper ($0.27 vs $9.04 per query) and 15× faster (63s vs 919s per query).”

- All three case studies (AlphaGenome, TISSUE, Scanpy) represent exceptionally well-documented projects (including one from the authors themselves) with active maintenance and clear tutorials, so they may not reflect the typical state of research code; evaluate Paper2Agent on papers with moderate documentation quality, older codebases, or partial implementations to characterize robustness.

Response: We thank the reviewer for this suggestion. To explicitly address robustness beyond exceptionally well-documented projects, our large-scale evaluation intentionally sampled 100 computational papers spanning a wide range of repository conditions, including moderate or minimal documentation, partial or research-grade implementations, and older or less-maintained codebases. These repositories were drawn at random from bioRxiv’s bioinformatics category and reflect the typical heterogeneity of research code in practice.

Under this setting, Paper2Agent successfully constructed executable agents for 74% of papers and achieved 91.2 ± 1.6% accuracy on execution-based benchmarks, demonstrating that its performance is not limited to highly curated or actively maintained projects. Failure cases were primarily attributable to missing code, unreleased data or models, or non-generalizable scripts, rather than documentation quality alone. These results show that Paper2Agent remains robust across realistic research codebases and that the three detailed case studies serve as illustrative examples rather than best-case outliers. As detailed above, we have revised our manuscript to reflect the new evaluations.

- The benchmarking sample size is limited to 15 tutorial-based and 15 novel queries per agent, which provides insufficient statistical power to claim 100% accuracy as generalizable performance; expand to at least 100 diverse queries per agent including intentionally challenging edge cases, and report confidence intervals and statistical significance tests.

Response: We thank the reviewer for this suggestion. To expand our benchmarking and include more challenging queries, we expanded our evaluation along four complementary axes: (i) 15 new open-ended queries for AlphaGenome that require multi-step reasoning and analytical strategies not explicitly demonstrated in the tutorials; (ii) 300 benchmark questions spanning 74 successfully agentified computational biology papers; (iii) 100 questions from non-computational papers that test synthesis across main text and supplementary materials; and (iv) additional computational papers from non-biology domains to assess generalization across scientific fields. Together, these evaluations substantially increase both the diversity and difficulty of the benchmark and directly address concerns about statistical power and generalizability.

For AlphaGenome, the new open-ended queries go beyond single tool calls with one correct numeric or categorical output. Instead, they require the agent to (1) independently formulate an analysis plan, (2) compose multiple tool calls (e.g., comparing variant effect predictions across tissues, integrating motif and QTL evidence, or evaluating multiple candidate variants), and (3) synthesize results into a biological conclusion. Example queries include: “Determine which variant is more likely to impact chromatin accessibility for the BACH2 gene in immune cells: chr6:90277329:G>T or chr6:90267049:G>A?”, which requires running and comparing ATAC-seq predictions across immune cell types for two candidate caQTL variants. Each question was evaluated across five independent runs and scored by two domain experts using a predefined rubric based on key entity matching (e.g., correct gene, variant, or biological conclusion). On this benchmark, Paper2Agent achieved 84.0 ± 2.7% accuracy, significantly outperforming Claude with repository access (65.3 ± 3.9%, paired t-test $p = 0.004$) and Biomni (76.0 ± 4.5%, $p = 0.035$).

In parallel, and as described above, we substantially expanded the evaluation for computational biology papers to 300 benchmark questions spanning 74 successfully agentified repositories, providing over an order of magnitude more queries than the original study. This benchmark includes tutorial-derived questions, structurally novel queries, and intentionally challenging edge cases such as missing inputs, alternative parameterizations, cross-method comparisons, and multi-step reasoning not explicitly demonstrated in tutorials. We report bootstrap standard errors

and confidence intervals across five independent runs. Under this evaluation, Paper2Agent achieved 91.2 ± 1.6% accuracy, significantly outperforming the baseline at 80.3 ± 2.3% ($p < 0.0001$, bootstrap test; 95% CI of difference: [5.8%, 15.8%]).

To assess generalization beyond biology, we further evaluated Paper2Agent on five non-biology computational papers spanning diverse scientific domains, including causal inference, mechanistic interpretability, natural language processing, computer vision, and tabular machine learning (TabPFN). Across 17 execution-based benchmark questions derived from these papers, Paper2Agent achieved 95.3% ± 1.9% average accuracy, demonstrating that the agentification pipeline generalizes beyond domain-specific assumptions and biology-centric tooling.

Finally, to evaluate performance beyond using tools in the paper, we curated 100 questions from 26 papers that require synthesis across main text and supplementary materials. On this benchmark, Paper2Agent achieved 89.0 ± 3.1% accuracy, significantly outperforming a strong baseline using Claude with browser-use capabilities and the paper URL (82.0 ± 3.8%, $p = 0.03$, bootstrap test), while being substantially more efficient in both cost and runtime.

Together, these results demonstrate that Paper2Agent generalizes beyond tutorial-level parameter variation, well-documented examples, and biology-specific domains. It maintains strong performance on open-ended, multi-step analytical tasks, large-scale heterogeneous computational benchmarks across disciplines, and non-computational papers requiring complex information synthesis, addressing concerns about robustness, scope, and generalizability. We have revised our manuscript to add those diverse benchmarks results.

- The "novel" queries for AlphaGenome appear to be structural variations of tutorial queries (e.g., different chromosomal positions, tissues, or modalities) rather than conceptually new analysis types, so they may not adequately test generalization beyond parameter variation; include queries requiring multi-step reasoning, tool composition, or approaches not directly demonstrated in tutorials.

Response: We thank the reviewer for the suggestion. We have expanded the AlphaGenome evaluation with 15 complex, open-ended queries that require multi-step reasoning and analytical strategies not explicitly demonstrated in the tutorials. Unlike the original benchmark, which tested individual tool calls with single correct numeric or categorical answers, these questions require the agent to (1) independently formulate an analysis plan, (2) compose multiple tool calls (e.g., running variant effect predictions across tissues, comparing two candidate variants, or interpreting motif and QTL evidence), and (3) synthesize results into a biological conclusion. Example queries include "Determine which variant is more likely to impact chromatin accessibility for the BACH2 gene in immune cells: chr6:90277329:G>T or chr6:90267049:G>A?", which requires comparing ATAC-seq predictions across immune cell types for two candidate caQTL variants.

Each question was run five times per method and scored by two human experts using a predefined rubric based on key entity matching (e.g., correct gene, variant, or biological conclusion). On this benchmark, Paper2Agent achieved 84.0 ± 2.7% accuracy, significantly outperforming both Claude + repository (65.3 ± 3.9%, paired t-test p=0.004) and Biomni (76.0 ± 4.5%, p=0.035). These results are consistent with performance observed in the 30 tutorial and original novel-query benchmarks, supporting robust generalization beyond parameter variation. We have added these results to the revised manuscript.

Revised Results section:

"We further evaluated the AlphaGenome agent on 15 open-ended queries reflecting realistic researcher usage, such as "Analyze the predicted accessibility and gene expression effects for chr22:45969257:G>A, associated with reduced bone mineral density. What is a likely mechanism of action and causal gene for this variant?". These queries require multi-step reasoning, tool composition, and analytical strategies not explicitly demonstrated in the tutorials. Paper2Agent still achieves superior performance (84.0% ± 2.7%) compared with Claude + Repo (65.3% ± 3.9%, $p = 4.3 \times 10^{-3}$) and Biomni (76.0% ± 4.5%, $p = 3.5 \times 10^{-2}$) across five independent runs, while also being more efficient (4.9 min, $0.37 per query) than Claude + Repo (8.0 min, $0.50 per query) and Biomni (10.0 min, no cost data available)."

- The paper claims tools are "validated" through iterative testing that verifies reproduction of tutorial outputs, but this constitutes training and testing on the same distribution, thereby creating a risk of overfitting where tools reproduce expected outputs for the wrong reasons; implement train/test splits using different tutorials for validation than for tool creation, and include adversarial examples designed to expose shallow pattern matching.

Response: We thank the reviewer for the suggestion. Paper2Agent addresses this concern at multiple layers, and we further strengthened the evaluation to explicitly test for such behavior.

First, in the original manuscript, generalization was evaluated using novel queries that were distinct from the tutorials used during tool construction. These queries, spanning AlphaGenome, Scanpy, and TISSUE, required the agent to apply extracted tools to new data rather than reproducing tutorial outputs. Specifically, they involved unseen genomic loci, tissues, or data modalities, ensuring that evaluation was performed on held-out inputs rather than the examples used for verification. Accordingly, tutorial-based testing served solely to ensure functional correctness of the extracted tools, while downstream performance was assessed on independent, non-tutorial queries.

Second, in the revised manuscript we further expanded this evaluation with open-ended queries that go beyond parameter variation. These questions require the agent to independently formulate an analysis plan, compose multiple tool calls, and synthesize results into a biological conclusion. Because these tasks are not directly demonstrated in any tutorial, they explicitly test whether the agent relies on executable semantics rather than reproducing expected outputs by pattern matching.

Third, to probe for potential shortcut learning or tutorial-specific memorization at scale, we performed a targeted inspection of the generated MCP server Python implementations across 100 randomly selected computational papers. Specifically, we examined whether the extracted tools contained tutorial-specific constants, fixed file paths, cached outputs, or dataset-dependent heuristics that would allow reproducing expected tutorial results without executing the intended computation. We did not find evidence of such hard-coded tutorial artifacts in the generated tool implementations, suggesting that Paper2Agent's tool layer is not simply copying tutorial outputs via shallow pattern matching. Moreover, we implemented a reviewer agent that automatically scans generated MCP server implementations to flag potential hard-coded values, fixed dataset paths, cached outputs, or tutorial-specific logic. This provides an additional automated check for implementation-level shortcut learning beyond manual inspection.

Together, these results demonstrate that tutorial-based validation serves only as a functional correctness check during tool construction, while generalization is evaluated on held-out, novel, and open-ended tasks. Across both targeted and large-scale evaluations, Paper2Agent exhibits behavior consistent with semantic execution on new inputs rather than overfitting to tutorial distributions.

We have revised our manuscript to reflect this.

Revised Results section:

"The generated MCP server implementations were further inspected for potential shortcut learning or tutorial-specific memorization. We did not find systematic evidence of hard-coded tutorial constants, fixed dataset paths, cached outputs, or dataset-specific heuristics."

- The manuscript states that tools which "repeatedly fail" are excluded from the MCP but does not report how many tools were attempted versus successfully retained, so the 100% accuracy claim may reflect survivorship bias; report the fraction of attempted tools that pass validation for each case study and provide failure case analysis characterizing why excluded tools could not be salvaged.

Response: We thank the reviewer for the suggestion. In our three case studies (AlphaGenome, scanpy, and TISSUE), all tools (100%) passed validation. We further added validation statistics for our large-scale benchmark across 100 computational biology papers.

Among the 74 successfully agentified papers, the agent initially proposed 599 tools, of which 593 passed validation testing (99.0% pass rate). The 6 excluded tools failed due to environment or infrastructure constraints (e.g., missing runtimes, version mismatches, strict input validation incompatible with synthetic tests, or excessive execution time). This high validation rate indicates that our approach does not suffer from survivorship bias. We have revised our manuscript to reflect this.

Revised Results section:

"Paper2Agent generated 22 AlphaGenome MCP tools, all of which passed automated validation, in around 45 minutes costing $14 on a personal laptop without human intervention."

"Paper2Agent generated 6 tools for the TISSUE MCP server, all of which passed automated validation, in 55 min costing $8, covering spatial gene expression prediction, prediction interval construction, and uncertainty-aware downstream analysis such as hypothesis testing, prediction, and dimensionality reduction."

"Paper2Agent generates 7 tools for this feature, all of which pass automated validation, in around 45 minutes costing $13 on a personal laptop."

"Among the 100 computational papers, 74 of 100 (74%) were successfully agentified with executable tools. Paper2Agent proposed 599 tools in total, of which 593 passed automated validation (99.0%)."

- The comparison with Claude Code + repository access does not account for the ~3 hours of upfront MCP creation time, so the reported 1.8–4.6× speedup is misleading without amortization analysis; report the number of queries needed before Paper2Agent becomes cost-effective compared to direct repository use, including API costs for both Claude and AlphaGenome.

Response: We appreciate this suggestion and have added an amortization analysis to the revised manuscript. The reviewer noted ~3 hours for upfront MCP creation; this referred to an earlier prototype. We have since upgraded our pipeline with a TypeScript-based orchestrator, reducing MCP creation time to 45 minutes.

Aggregated across all three benchmarks (Novel, Tutorial, open-ended; 225 queries total), the MCP-based agent costs $0.289/query versus $0.349/query for Claude Code + repository access, a saving of $0.060/query, while median latency drops from 141s to 63s (2.2×).

The time break-even is reached after approximately 19–33 queries (2,580s upfront / 78–135s saved per query). The cost break-even requires approximately 245 queries ($14.67 / $0.060 per query). While the cost break-even is ~245 queries, these MCP tools are a one-time community investment. Once a lab or a journal agentifies a paper, every subsequent researcher globally benefits from the $0.06/query saving and the 2x speedup; in practice, the primary benefit is the 2× latency reduction per query, which compounds with interactive use, and improved accuracy. We have added those to the Results section in the revised manuscript.

Revised Results section:

"The AlphaGenome agent also consistently outperformed both other agent systems in terms of compute efficiency, showing a median decrease in run time for the tutorial-based benchmark of

1.9x and 3.1x compared to Claude + Repo and Biomni, respectively. For the novel benchmark, we observed a median run time improvement of 2.9x and 3.8x, respectively. These per-query speedups do not account for the one-time upfront cost of MCP creation. The time break-even is reached after approximately 19--33 queries. While the cost break-even is around 245 queries, these MCP tools are a one-time community investment. Once a lab or a journal agentifies a paper, every subsequent researcher globally benefits from the $0.06/query saving and the 2x speedup. These results highlight the improved reliability and efficiency achieved by Paper2Agent-generated agents compared to other agentic systems."

- The paper positions MCP as central but does not compare with simpler alternatives such as well-structured Markdown skill files (broadly applicable but formalized recently with Claude's new Skills feature), so the necessity and added value of the MCP architecture remain undemonstrated; the authors should include a controlled comparison where AlphaGenome functionality is exposed via a Markdown skill file versus an MCP server, testing whether Claude Code achieves similar performance with the simpler approach.

Response:

We thank the reviewer for this comment. We conducted a controlled comparison where AlphaGenome functionality is exposed via a Markdown skill file (Claude Code's Skills feature) versus an MCP server, using the same base model (Claude Sonnet 4), the same 30 benchmark questions (15 tutorial + 15 novel), and 5 independent runs per question.

The Markdown skill file approach achieved an accuracy of 70.7% ± 4.5% on tutorial questions and 76.0% ± 2.7% on novel questions. In comparison, the MCP-based approach achieved 98.7% ± 1.3% on tutorial and 100.0% ± 0.0% on novel questions.

We attribute this difference to several architectural advantages of MCP generated by Paper2Agent. First, MCP tools execute API calls directly on the server side, whereas skill files require the agent to write and execute Python scripts to interact with the API, introducing additional points of failure. Second, MCP provides structured input and output schemas that constrain the agent's tool invocations, reducing hallucinated or malformed calls. Third, Paper2Agent includes a dedicated test-verifier-improver agent that automatically validates each extracted tool against tutorial examples and iteratively repairs failures, ensuring that the deployed MCP server is functionally correct and robust. Finally, skill files and MCP servers are designed for somewhat different purposes: skill files primarily encode high-level task procedures (i.e., how to accomplish a task by invoking a sequence of tools), which is conceptually similar to the role of MCP prompts in Paper2Agent, whereas MCP servers focus on exposing reliable, executable tools with well-defined interfaces. Together, these results demonstrate that while Markdown skill files offer a simpler setup, the MCP architecture provides meaningful performance gains that justify its adoption, particularly for scientific APIs where precise parameter handling and reliable execution are critical. We have revised the Results and Discussion sections accordingly in the revised manuscript.

Revised Results section:

“Replacing MCP tools generated by Paper2Agent with Markdown skill files also degraded performance. On 30 AlphaGenome benchmark questions (tutorial and novel), the skill-file approach achieved 73.3 ± 2.6% accuracy (mean ± SE), compared to 99.3% ± 0.7% with MCP-based tools across five independent runs. We attribute this gap to two factors. First, MCP enforces structured schemas and a constrained action space, reducing tool invocation errors. Second, the test-verifier-improver agent automatically validates and iteratively repairs tools against tutorial references, ensuring functional correctness prior to deployment.”

Revised Discussion section:

”Our comparison also highlights a useful design distinction: skill files encode high-level task procedures describing how to accomplish a goal, whereas MCP servers expose executable tools with well-defined interfaces. These roles are complementary — skill files can serve as orchestration layers that coordinate calls to underlying MCP tools, combining procedural flexibility with execution reliability. Indeed, Paper2Agent already adopts this pattern: each generated agent includes MCP prompts that guide the agent on when and how to invoke the underlying MCP tools, functioning analogously to skill files built atop a robust tool layer.“

- The entire implementation relies on Claude Code (claude-sonnet-4-20250514) and the multi-agent orchestration is built specifically for this platform, so generalizability to other AI systems is unknown and the framework may represent a Claude-specific application rather than a reusable contribution; test Paper2Agent with at least one non-Claude agent (e.g., Codex CLI or even an open model via OpenCode eg Qwen3-coder) and report whether comparable conversion quality can be achieved.

Response: We thank the reviewer for the suggestion. We have added an additional evaluation of Paper2Agent using an alternative agent backend based on OpenCode. We ran the same conversion pipeline on AlphaGenome and compared the resulting MCP server and downstream task performance to those generated by Claude Code.

Using OpenCode, Paper2Agent successfully extracted 20 tools from AlphaGenome within approximately 65 minutes. The extracted tools were highly similar in structure and functionality to those produced by Claude Code. We further benchmarked the AlphaGenome agent using the MCP server generated by OpenCode and found that it achieved 98.7% ± 1.3% accuracy on tutorial queries and 97.3% ± 1.6% accuracy on novel queries, comparable to the performance obtained with Claude Code.

These results suggest that while our implementation benefits from Claude Code’s strong agentic tooling, the core Paper2Agent framework is not inherently platform-specific and can be adapted to alternative agent scaffolding with comparable conversion quality. We have revised the results section to add these new results.

Revised Results section:

“Using the OpenCode backend, Paper2Agent extracted comparable tools and achieved 98.7 ± 1.3% accuracy on tutorial queries and 97.3 ± 1.6% on novel queries on the AlphaGenome benchmarks, indicating that conversion quality is not specific to the Claude Code backend.”

- The comparison with Claude Code + repository is confounded because the baseline ("Claude + Repo") appears to receive only the code repository and query, while Paper2Agent receives the repository plus the paper manuscript, structured tool definitions, and pre-validated workflows, so it remains unclear whether performance gains stem from the MCP infrastructure or simply from providing the manuscript text as context; include a comparison where Claude Code receives both the paper PDF and repository (matching Paper2Agent's information access) to establish whether the MCP format and automated conversion provide measurable benefit beyond giving Claude Code equivalent information directly.

Response: We thank the reviewer for raising this point and would like to clarify the benchmarking setup. In the reported benchmarks, Paper2Agent does not have access to the paper manuscript text. During evaluation, the agent interacts only with the generated MCP server, which exposes a fixed set of executable tools and minimal metadata produced during the automated conversion process. The paper PDF and repository contents are not visible to the agent when answering benchmark queries. We have added the clarifications in the revised paper.

Revised Results section:

“Next, we benchmarked the Paper2Agent-generated AlphaGenome agent in producing numerical and qualitative results and figures relative to human experts configuring and running the code manually. We also compared the AlphaGenome agent to two other agentic systems: 1) Claude Code with access to the AlphaGenome repo (referred to as Claude + Repo) and 2) Biomni. The Paper2Agent-generated agent does not have access to the paper manuscript or raw repository during evaluation.”

- The manuscript does not evaluate the contribution of individual sub-agents in the multi-agent conversion pipeline (environment-manager, tutorial-scanner, tool-extractor-implementor, test-verifier-improver), so it remains unclear which components are essential versus dispensable; conduct ablation studies removing or simplifying each sub-agent to assess whether the full multi-agent architecture is necessary or whether simpler approaches (e.g., a monolithic conversion script or manual curation of specific steps) could achieve comparable MCP quality with less complexity.

Response: We thank the reviewer for this suggestion. The multi-agent design was originally chosen for three main reasons: (i) to prevent context-window saturation during long-horizon code analysis (we observed that collapsing all steps into a single agent leads to severe context blowup and degraded tool quality), (ii) to enable parallel execution of independent sub-tasks, including environment setup, tutorial discovery, tool extraction, and verification, (iii) to ensure the implemented MCP Python script is correct through the test-verifier-improver.

To evaluate whether this modular design is necessary, we conducted an ablation study using the AlphaGenome repository with three simplified variants of the pipeline. Each condition was run three times to estimate stability and success rates: (1) Monolithic agent: all sub-tasks are executed sequentially within a single agent and a single context window (200K tokens) (2) Non-parallel multi-agent: the same sub-agents are used, but forced to run sequentially rather than in parallel. (3) No test-verifier-improver agent.

Compared to the full multi-agent pipeline, the monolithic agent failed to generate a valid MCP server Python script in all three runs. Failure analysis revealed that the agent was unable to maintain coherent long-horizon execution: in run 1, the agent prematurely requested to terminate the workflow ("Would you like me to continue with Phase 3?"); in run 2, it recommended skipping critical steps entirely ("Tool extraction is not feasible"); and in run 3, it explicitly admitted failure ("NOT COMPLETED… due to scope and complexity"), achieving only partial completion (~40%).

In contrast, the non-parallel multi-agent variant was able to generate a usable MCP server across all three runs, but required substantially longer wall-clock time. Specifically, this variant took 2.2–2.4x longer (93m13s – 104m44s per run) compared to the parallel multi-agent system (42m23s – 43m39s per run). Together, these results indicate that both modularization and parallel execution are important for reliable and efficient agentification. We have revised the results section to add these new results.

Finally, removing the test-verifier-improver agent led to unstable tool quality: although MCP servers were still generated, several extracted tools failed to reproduce tutorial outputs or produced silent numerical errors that were detectable only through manual inspection. On our tutorial and novel benchmarks, accuracy dropped to 69.3 ± 4.5% and 84.0 ± 2.7% respectively, compared to 98.7% ± 1.3% and 100.0% ± 0.0% with the full system, demonstrating that automated verification is essential for ensuring tool correctness, not merely a convenience. We have added those new results to our revised manuscript:

Revised Results section:

"Next, we conducted ablation studies to evaluate the benefit of Paper2Agent's architectural design choices, including the multi-agent orchestration, parallel execution, automated test-driven verification, the MCP tool interface, and agent scaffolding. We performed these studies on the AlphaGenome repository by comparing the full Paper2Agent system with several simplified variants: (1) a monolithic agent, where all sub-tasks are executed within a single context window; (2) a non-parallel multi-agent system, where sub-agents run sequentially; (3) a system without the test-verifier-improver agent; (4) an alternative implementation using Markdown skill files instead of MCP tools; and (5) an alternative coding agent (OpenCode) to test scaffolding robustness.

The monolithic agent failed to generate a valid MCP server in all three runs due to context management and planning failures. The non-parallel multi-agent variant succeeded but required 2.2–2.4× longer runtime. Removing the test-verifier-improver agent led to unstable tool quality:

accuracy dropped to 69.3 ± 4.5% and 84.0 ± 2.7% on tutorial and novel benchmarks, respectively, compared to 98.7% ± 1.3% and 100.0% ± 0.0% with the full system."

- The ADHD case study claims the AI co-scientist "identified new splicing variant associated with ADHD risk," but rs1626703 was already in the fine-mapping credible set from the original GWAS paper and the AlphaGenome predictions are computational rather than experimentally validated, so this represents hypothesis prioritization rather than discovery; revise all "discovery" and "identified" language to "prioritized" or "generated computational hypothesis," especially in the abstract which is particularly misleading for a high-profile venue.

Response: We thank the reviewer for the suggestions. We have modified the language to "prioritized" for the ADHD case study. To complement this and demonstrate downstream biological validation, we have added an additional case study in which the AI co-scientist's top computational hypothesis is subsequently supported by independent experimental datasets.

In a second application, we used three paper agents generated by Paper2Agent, AlphaGenome, MPRA-coupled scCRISPRi, and CD4+ T cell Perturb-seq, to prioritize and validate the causal gene for a psoriasis-associated variant. The AlphaGenome agent predicted GPR137 as the top affected gene at the rs887314 locus in CD4+ T cells (RNA-seq quantile score = 0.997), ranking above other nearby candidate genes. To validate this prediction, we instructed the Paper2Agent co-scientist to cross-reference the AlphaGenome prediction with experimental data from two additional paper agents: an MPRA-coupled scCRISPRi screen measuring downstream gene expression changes following CRE perturbation, and a genome-scale Perturb-seq dataset profiling transcriptional consequences of gene knockdowns in primary human CD4+ T cells.

After autonomously inspecting the main manuscript, analyzing the supplementary tables from the MPRA-coupled scCRISPRi paper, and examining the differential expression summary statistics from the CD4+ T cell Perturb-seq paper, the AI co-scientist proposed 10 candidate strategies to validate this gene. From these, the human researcher selected signature correlation analysis to test whether the CRE perturbation signature matched any gene knockdown signatures. For each of the five top-ranked candidate genes with available knockdown data, the agent correlated downstream gene expression changes caused by perturbing the rs887314 cis-regulatory element with those caused by knocking down each candidate gene across three culture conditions (Rest, Stim8hr, Stim48hr). Only GPR137 knockdown showed significant concordance with the CRE perturbation signature (Spearman rho = 0.613, P = 3.79e-3 at Stim8hr; rho = 0.630, P = 4.71e-3 at Stim48hr; FDR < 0.05), while BAD knockdown and three other candidate genes showed no significant correlation in any condition. These experimental results support GPR137 as the likely causal gene mediating the effect of rs887314 on psoriasis risk.

We have revised the manuscript to add this discovery.

Revised Results section:

“In a second application, we used three agents generated by Paper2Agent, AlphaGenome, MPRA-coupled scCRISPRi, and CD4+ T cells Perturb-seq, to identify and validate the causal gene for a psoriasis-associated variant. The AlphaGenome agent predicted GPR137 as the top affected gene at the rs887314 locus in CD4+ T cells (RNA-seq quantile score = 0.997), ranking above other nearby genes. To validate this prediction, we instructed the AI co-scientist to cross-reference the AlphaGenome prediction with experimental data from two additional paper agents: an MPRA-coupled scCRISPRi screen that measured downstream gene expression changes upon cis-regulatory element (CRE) perturbation, and a genome-wide Perturb-seq dataset that profiled transcriptional consequences of gene knockdowns in primary human CD4+ T cells.

After autonomously inspecting the main manuscript, analyzing the supplementary tables from the MPRA-coupled scCRISPRi paper, and examining the differential expression summary statistics from the CD4+ T cell Perturb-seq paper, the AI co-scientist proposed 10 candidate strategies to validate this gene. From these, the human researcher selected signature correlation analysis to test whether the CRE perturbation signature matched any gene knockdown signatures. For each of the five top-ranked candidate genes with available knockdown data, the agent correlated the downstream gene expression changes caused by perturbing the rs887314 cis-regulatory element with those caused by knocking down each candidate gene across three culture conditions (Rest, Stim8hr, Stim48hr). Only GPR137 knockdown showed significant concordance with the CRE perturbation signature (Spearman r = 0.613, P = 3.79 × 10^-3 at Stim8hr; rho = 0.630, P = 4.71 × 10^-3 at Stim48hr; FDR < 0.05), while BAD knockdown and three other top-ranked candidate genes showed no significant correlation in any condition. These results demonstrate that Paper2Agent enables the autonomous integration of computational predictions with independent experimental validation, supporting GPR137 as the likely causal gene mediating psoriasis risk at the rs887314 locus.”

- The AlphaGenome agent prioritizes SORT1 (quantile score 0.99982) over the original paper's CELSR2 and PSRC1 (both 0.99998), but these quantile scores are functionally indistinguishable (all in top 0.02% of variants, differing by only 0.016 percentile points) and all three genes show significant eQTL associations in GTEx liver tissue, thereby illustrating that AlphaGenome provides no discriminatory power at complex GWAS loci where multiple nearby genes have extreme predictions; frame this result as "rapid generation of alternative hypotheses for human evaluation" rather than improved inference, since the agent essentially applied functional reasoning to break ties between computationally equivalent predictions.

Response: We thank the reviewer for this clarification. In the revised manuscript, we have reframed this result to emphasize “rapid generation of alternative hypotheses for human evaluation” rather than improved inference. We now explicitly state that at complex loci where multiple nearby genes exhibit extreme functional predictions and significant eQTL associations, the agent's primary value lies in its ability to break ties through systematic biological reasoning. While the AlphaGenome model provides the raw scoring, the agent layer autonomously synthesizes additional evidence, such as *SORT1*’s direct involvement in LDL/VLDL secretion, to prioritize a specific candidate for further investigation. This process demonstrates how Paper2Agent can surface plausible mechanisms that may have been deprioritized in the original

narrative, thereby facilitating a more comprehensive hypothesis-reassessment process. We have revised our manuscript to reflect this:

Revised manuscript:

"This discrepancy highlights a key strength of Paper2Agent: with a single prompt, users can re-evaluate published conclusions using independent model-based evidence, without the need to design new analysis pipelines."

- The manuscript does not report inter-rater reliability for ground truth validation of novel queries, which relies on "manual execution of the original AlphaGenome code" by human experts, so the objectivity and reproducibility of the 100% accuracy claim remain uncertain; include multiple independent expert raters and report inter-rater agreement statistics.

Response: We thank the reviewer for this suggestion. In the revised manuscript, we strengthened the ground truth validation protocol for novel queries by incorporating multiple independent expert raters. Each query was independently executed and evaluated by two domain experts using the original AlphaGenome codebase and standardized evaluation criteria.

Inter-rater agreement was near perfect. For tutorial queries (n = 15), raters produced identical labels (100% agreement). For novel queries (n = 15), raters disagreed on 1 query (6.7%), which was resolved through discussion using the predefined evaluation rubric. Based on this predefined rubric (defined prior to evaluation), both candidate responses were considered correct because they produced numerically and biologically equivalent conclusions, reflecting multiple acceptable analysis paths rather than uncertainty in the ground truth.

Overall inter-rater agreement across all 30 queries was 96.7% (29/30 exact agreement). We now report the full multi-rater protocol and agreement statistics in the revised manuscript.

Revised Results section:

"*These reported performance improvements were validated by two independent human experts who manually executed the original AlphaGenome code following predefined evaluation rubrics (inter-rater agreement: 96.7%)"*

- The paper does not characterize agent failure modes such as handling of ambiguous queries, requests outside the paper's scope, or malformed inputs, so users cannot distinguish agent limitations from paper limitations and false positive rates are unknown; systematically evaluate a set of out-of-scope or adversarial queries and report how often the agent incorrectly claims capability versus appropriately declines.

Response: We thank the reviewer for this suggestion. To explicitly characterize agent failure modes and false positive behavior, we performed two out-of-scope query evaluations using non-matching paper–question pairs.

Specifically, we constructed an adversarial benchmark by randomly permuting questions and papers across 26 non-computational papers (13 bioRxiv and 13 Nature papers). This ensures that each question is paired with a paper that does not contain the information required to answer it. We evaluated Paper2Agent (with sonnet4.5) under two prompt conditions:

(1) **Explicit rejection instruction**: The agent was prompted with: "You are analyzing a scientific paper. Answer this question based on the files in the current directory. If the question is not related to the files, say 'I don't know'." Under this setting, Paper2Agent achieved a 100% correct rejection rate (26/26), consistently declining to answer out-of-scope questions rather than hallucinating plausible but unsupported answers.

(2) **No explicit rejection instruction**: To test whether the agent's behavior depends on being explicitly told to decline, we removed the "If the question is not related to the files, say 'I don't know'" instruction and re-ran the same benchmark. Paper2Agent still achieved a 100% correct rejection rate (26/26), indicating that the structured resource layer itself provides sufficient grounding for the agent to recognize when a question falls outside the scope of the available paper, without requiring explicit refusal prompting.

Together, these results demonstrate that Paper2Agent reliably distinguishes in-scope from out-of-scope queries and does not fabricate answers when information is unavailable, regardless of whether refusal behavior is explicitly instructed. We have added these results to the revised paper.

Revised Results section:

“To understand whether the agent will reject out-of-scope queries, we constructed an adversarial out-of-scope benchmark by randomly permuting paper–question pairs across the 26 data and discovery papers. Paper2Agent achieved a 100% correct rejection rate under both prompted and unprompted conditions, reliably declining to answer mismatched paper–question pairs even without explicit instructions to do so.”

- All case studies are from computational biology and bioinformatics, so generalizability to other scientific domains (chemistry, physics, machine learning, social sciences, wet-lab experimental papers) is unsupported; I would suggest demonstrating Paper2Agent on at least two papers from non-biology fields or explicitly scope the contribution to computational biology with empirical justification for this limitation.

Response: We thank the reviewer for this suggestion. In the revised manuscript, we address generalizability through two new analyses spanning non-biology domains and non-computational paper types (including wet-lab experimental papers).

Non-biology case studies (5 repositories, 17 benchmark questions). We extended our evaluation to five repositories outside computational biology: GRF (causal inference/econometrics), SAELens (mechanistic interpretability), Binoculars (LLM-generated text detection/NLP), SAM2 (computer vision), and TabPFN (tabular ML). For each, we ran

Paper2Agent end-to-end to generate an AI agent and evaluated it on execution-based benchmark queries derived from the papers' tutorials and documentation. Across 5 independent runs on 17 questions, Paper2Agent achieves 95.3% ± 1.9% (SE) average accuracy, demonstrating robust generalization to econometrics, machine learning, NLP, and computer vision.

Non-computational papers (26 papers, 100 questions). Paper2Agent also enables retrieval-augmented question answering based on its structured resource layer (manuscript text, supplementary tables, and metadata). We curated 100 questions from 13 bioRxiv preprints and 13 Nature articles — including wet-lab experimental papers — requiring synthesis across main text and supplementary materials (e.g., "The paper reports results using Pearson correlation; reanalyze the conclusions using Spearman correlation"). Paper2Agent achieved 89.0 ± 3.1% accuracy, significantly outperforming a strong baseline of Claude with browser-use capabilities and the paper URL (82.0 ± 3.8%; p = 0.03, bootstrap test). Paper2Agent was also 34× cheaper ($0.27 vs. $9.04 per question) and 15× faster (63 s vs. 919 s per question).

Together, these results demonstrate that Paper2Agent generalizes robustly, both to non-biology domains (econometrics, ML, NLP, computer vision) and to non-computational paper types including wet-lab experimental work. We have added those results to the Results section of the paper.

Revised results section:

"We extend our evaluation to include five non-biology case studies including diverse scientific domains: causal inference and econometrics (grf), mechanistic interpretability (SAELens), natural language processing (Binoculars), computer vision (SAM2), and tabular machine learning (TabPFN). For each repository, we ran Paper2Agent end-to-end to generate an AI agent, then evaluated it on execution-based benchmarking queries derived from the papers' tutorials and documentation. The benchmark includes 17 questions covering tasks such as causal forest training, sparse autoencoder analysis, text detection, image segmentation, and tabular classification. Overall, Paper2Agent achieves 95.3% ± 1.9% (SE) average accuracy across five independent runs, demonstrating robust generalization beyond computational biology."

"Even when executable tools cannot be constructed, Paper2Agent still provides value through its structured resource layer. Across 26 non-computational papers (13 bioRxiv and 13 Nature papers), we curated 100 synthesis-based benchmark questions spanning main text and supplementary materials. Paper2Agent achieved 89.0 ± 3.1% accuracy, significantly outperforming a strong Claude browser-use baseline (82.0 ± 3.8%, p = 0.03), while being 34× cheaper ($0.27 vs $9.04 per query) and 15× faster (63s vs 919s per query)."

- The manuscript defers critical algorithmic details to the GitHub repository, including how tutorials are selected from repositories, how tools are extracted from tutorial code, what constitutes "repeated failure" for tool exclusion, and how MCP prompts are constructed, thereby

making it difficult to evaluate reproducibility of Paper2Agent itself; perhaps move sufficient detail into Extended Methods with pseudocode for key procedures so that the paper is self-contained enough for independent reimplementation.

Response: We thank the reviewer for the suggestion. In the revised manuscript, we have expanded the Extended Methods and Supplementary Note. Specifically, we now provide step-by-step prompts and pseudocode for (i) tutorial identification and filtering, (ii) tool extraction and interface synthesis from tutorial code, (iii) the iterative test–verify–improve loop, including the definition of "repeated failure" and pruning criteria, and (iv) MCP prompt construction and resource packaging. We also clarify all inputs, outputs, and stopping conditions for each sub-agent.

- Statistical rigor is lacking, notably in Figure 2C which compares computational efficiency across agents but reports only median speedup without confidence intervals, significance tests, or even individual data points despite small sample sizes (15 queries per benchmark), so the reliability of claimed performance improvements remains uncertain; apply statistical tests (e.g., Wilcoxon signed-rank for paired runtime comparisons) and report confidence intervals to establish whether observed differences are statistically significant, and note that the predominance of qualitative schematic figures over quantitative analysis figures throughout the manuscript is unusual for a methods paper and limits rigorous evaluation of performance claims.

Response: We thank the reviewer for this suggestion. We have added the analysis that ran each method five times [5 runs × (15 + 15) questions = 150 evaluations per method] and reported mean accuracy ± SE across runs for the AlphaGenome.

Method	Tutorial tasks (mean ± SE)	Novel tasks (mean ± SE)
AlphaGenome Agent	98.7% ± 1.3%	100.0% ± 0.0%
Claude + repo	82.7% ± 3.4%	78.7% ± 4.4%
Biomni	37.3% ± 4.0%	56.0% ± 3.4%

To assess whether AlphaGenome agent significantly outperforms the baselines, we conducted one-sided paired t-tests on per-run accuracy. AlphaGenome agent significantly outperformed both Claude + repo (tutorial: $p = 8.1 \times 10^{-3}$; novel: $p = 4.2 \times 10^{-3}$) and Biomni (tutorial: $p = 4.7 \times 10^{-5}$; novel: $p = 1.0 \times 10^{-4}$). This is consistent with the original conclusion of the paper showing that the AlphaGenome agent generated by Paper2Agent outperforms alternative approaches. For other newly added benchmark results, we have also added the mean ± SE for the reported accuracy. We have added these results to the Results section of the revised manuscript.

Revised Method section:

"Across five independent runs, the AlphaGenome agent achieved 98.7% ± 1.3% accuracy on these queries, significantly higher than Claude + Repo (82.7% ± 3.4%, $p = 8.1 \times 10^{-3}$) and Biomni (37.3% ± 4.0%, $p = 4.7 \times 10^{-5}$)."

"Across five independent runs, the AlphaGenome agent achieved 100.0% ± 0.0% accuracy, as verified by manual execution of the original AlphaGenome code and evaluated by predefined rubrics. This is significantly higher than Claude + Repo (78.7% ± 4.4%, $p = 4.2 \times 10^{-3}$) and Biomni (56.0% ± 3.4%, $p = 1.0 \times 10^{-4}$)."

- The paper does not discuss maintenance burden as papers and codebases evolve (e.g., when AlphaGenome API changes or dependencies break), so the sustainability and economic model for agent ecosystems remain unaddressed; I suggest providing a concrete maintenance plan and discussing who bears responsibility for keeping agents synchronized with upstream changes.

Response: We thank the reviewer for this important point. In the revised manuscript, we now describe an automated maintenance loop built on agent-based code execution frameworks such as Claude Code Web and Jules.

Specifically, each MCP server can be paired with a lightweight maintenance agent implemented using Claude Code Web or Jules that periodically, or when upstream repositories change, re-runs the original validation tests in a clean environment. When failures are detected due to API changes, dependency drift, or missing artifacts, the agent attempts automated repairs (e.g., updating dependency pins, regenerating wrappers, or modifying execution paths) and re-validates before redeployment. If automated repair fails, the issue is flagged for human review rather than silently propagating errors.

We have added a new subsection in the Discussion outlining this maintenance workflow, including the division of responsibility: initial agentification is performed by method authors or curators, while routine re-validation and minor repairs are handled automatically by the maintenance agent via Claude Code Web or Jules, with human intervention reserved for substantive upstream changes.

Revised Discussion section:

"Although Paper2Agent focuses on initial agentification, long-term usability requires mechanisms to maintain agents as underlying codebases and dependencies evolve. To address this, each MCP server can be paired with a maintenance agent implemented using agentic code execution frameworks such as Claude Code Web or Jules, which periodically or upon upstream changes re-runs validation tests in an isolated environment. When failures occur due to API changes or dependency drift, the agent attempts automated repair and re-validation, and flags unresolved issues for human review."

- The manuscript does not provide a cost analysis including Claude API fees, domain-specific API costs (e.g., AlphaGenome), compute resources for MCP creation, and deployment hosting

fees, so the practical feasibility for typical research groups is uncertain; report total costs for creating each case study MCP and estimate ongoing costs for query execution and maintenance.

Response: We thank the reviewer for this suggestion. In the revised manuscript, we added an explicit cost analysis covering (i) one-time MCP creation costs (LLM API usage, compute time, and deployment) and (ii) recurring per-query execution costs (LLM calls, optional domain APIs such as AlphaGenome, and runtime).

In our large-scale evaluation across 100 computational papers, successfully agentified papers required $14.99 on average (2.4 hours) to build an MCP server, while failed attempts typically terminated earlier ($9.03; 1.7 hours). At query time, Paper2Agent answered benchmark questions at $0.20 per query in 1.6 minutes on average, compared to $0.38 per query in 4.3 minutes for the baseline. For the case studies, we now report explicit end-to-end MCP creation cost and runtime. The AlphaGenome MCP required approximately 45 minutes with $14 to construct end-to-end. The Scanpy MCP required approximately 45 minutes with $13 total LLM cost, and the TISSUE MCP required approximately 55 minutes with $8 total LLM cost. For 26 non-computational data and discovery papers, Paper2Agent answered synthesis-based queries at approximately $0.27 per query with an average runtime of 63 seconds, compared to $9.04 per query and 919 seconds for a browser-use baseline.

Regarding domain-specific and infrastructure costs, the AlphaGenome API is currently free to use. MCP deployment on Hugging Face Spaces is free for CPU-only MCP servers, which is sufficient for the majority of paper agents in this study.

We have revised our paper to add those results.

Revised Results section:

“We focus on Scanpy’s most common use case: preprocessing and clustering single-cell data. Paper2Agent generates 7 tools for this feature in around 45 minutes costing $13”

“Paper2Agent generated 22 AlphaGenome MCP tools in around 45 minutes costing $14 on a personal laptop without human intervention, comprehensively covering its methodological innovations.”

“Paper2Agent generated 6 tools for the TISSUE MCP server in 55 min costing $8.”

“We evaluated these 74 paper agents on 300 benchmark questions derived from GitHub tutorials. Paper2Agent achieved 91.2 ± 1.6% accuracy (mean ± bootstrap SE), significantly outperforming the baseline (Claude Code with direct repository access) at 80.3 ± 2.3% ($p < 0.0001$, bootstrap test; 95% CI of difference: [5.8%, 15.8%]). Paper2Agent answered benchmark questions at an average cost of $0.20 per query in 1.6 minutes, compared to $0.38

per query in 4.3 minutes for the baseline, corresponding to a 1.9× cost reduction and 2.7x speedup."

"Even when executable tools cannot be constructed, Paper2Agent still provides value through its structured resource layer. Across 26 non-computational papers (13 bioRxiv and 13 Nature papers), we curated 100 synthesis-based benchmark questions spanning main text and supplementary materials. Paper2Agent achieved 89.0 ± 3.1% accuracy, significantly outperforming a strong Claude browser-use baseline (82.0 ± 3.8%, p = 0.03), while being 34× cheaper ($0.27 vs $9.04 per query) and 15× faster (63s vs 919s per query)."

- The Scanpy MCP prompts encode a "standard" preprocessing and clustering pipeline that may not suit all single-cell datasets, and while the prompt instructs the agent to "inspect the data first," there is no validation that appropriate parameter adjustments actually occur, thereby creating a risk of cargo-cult analysis where steps are followed without domain-appropriate customization; please include examples demonstrating that the Scanpy agent adapts pipeline parameters based on dataset characteristics (e.g., mitochondrial percentage thresholds, resolution parameters).

Response: We thank the reviewer for this suggestion. In the revised manuscript, we now include data-adaptive scanpy case studies demonstrating that the agent adjusts key preprocessing and clustering parameters based on dataset characteristics rather than blindly following a fixed workflow. Specifically, we applied the pipeline to four distinct datasets and observed the following adaptive behaviors:

1. Organism-specific mitochondrial gene detection:
 - For human PBMC datasets (pbmc_1k_v3, pbmc_1k_v2), the agent correctly used the MT- prefix for mitochondrial genes
 - For the mouse hematopoiesis dataset (Paul et al. 2015), the agent automatically switched to the Mt- prefix, recognizing the organism from gene nomenclature

2. Adaptive highly variable gene selection based on gene panel size:
 - For full transcriptome datasets (pbmc_1k_v3: 15,247 genes; pbmc_1k_v2: 33,538 genes), the agent selected the standard 2,000 highly variable genes (HVGs).
 - For the Paul15 dataset with only 3,451 genes (a targeted panel), the agent explicitly reasoned: "Since this dataset has only 3,451 genes (targeted panel), I'll select fewer HVGs" and reduced the selection to 1,000 genes to maintain an appropriate ratio

3. Tissue-specific marker gene selection:
 - For PBMC datasets, the agent applied canonical immune cell markers (CD3D, CD3E, MS4A1, CD14, NKG7, etc.)
 - For the Paul15 bone marrow hematopoiesis dataset, the agent automatically switched to hematopoietic lineage markers (Gata1, Klf1 for erythroid; Elane, Mpo for granulocytes; Pf4, Itga2b for megakaryocytes; Kit, Cd34 for HSC/MPP)

These examples demonstrate that the Scanpy agent performs genuine data inspection and makes domain-appropriate parameter adjustments rather than executing a rigid, one-size-fits-all pipeline. We have added those results in the revised manuscript.

Revised Results section:

"We further applied it to diverse single-cell datasets and observed that the agent adaptively adjusts parameters based on data characteristics: selecting organism-appropriate mitochondrial gene prefixes, reducing HVG counts for targeted gene panels, and applying tissue-specific marker genes for cell type annotation."

- The paper briefly mentions "security defaults" for MCP servers but provides no threat model, vulnerability analysis, or discussion of what happens if malicious code in a paper's repository gets exposed through an MCP tool, so security implications remain unaddressed; maybe discuss intellectual property considerations when third parties create agents from published papers, and clarify attribution chains if agent-generated hypotheses lead to downstream discoveries.

Response: We thank the reviewer for raising these important points. In the revised manuscript, we have added a section in the Discussion. This section outlines a threat model for MCP servers (including risks from malicious or compromised repositories), describes sandboxing and permission defaults, and discusses intellectual property and attribution considerations when third parties create agents from published work. These additions clarify the security boundaries and ethical responsibilities associated with Paper2Agent.

Revised discussion section:

"Exposing research code as executable agents also introduces security, intellectual property, and attribution challenges. MCP servers may inherit vulnerabilities from their source papers and associated code repositories; tools are therefore executed in sandboxed environments with restricted permissions and retain traceable links to their source code for provenance. MCP servers remain subject to the original licenses of the underlying code, and third-party distribution should preserve authorship and usage constraints. When agent-generated analyses contribute to downstream discoveries, we recommend treating paper-derived agents as computational instruments, with credit assigned to the original method developers, data generators, and human investigators who designed the study."

- The paper does not report whether Paper2Agent inherits bugs from the original codebase, so if a paper's implementation contains errors, these may be propagated and potentially amplified in the generated MCP tools; maybe discuss how Paper2Agent handles codebases with known issues and whether the validation process can detect inherited bugs versus only confirming consistency with potentially flawed tutorial outputs.

Response: We thank the reviewer for this suggestion. Because repositories with known, well-documented issues are difficult to identify systematically, we evaluated Paper2Agent by manually injecting common execution-level errors into three representative repositories spanning different programming paradigms: AlphaGenome (Python-based), POP-TOOLS (command-line–based), and mlearner (R-based).

For each repository, we considered four adversarial configurations that simulate realistic implementation failures: (1) removal of required dependencies from environment specifications (e.g., requirements.txt), (2) modification of input file paths to nonexistent locations, (3) insertion of cell-level execution errors into notebooks or tutorials, and (4) replacement of API calls with deprecated equivalents when applicable. This resulted in a total of 12 adversarial case studies. We applied Paper2Agent to each configuration and recorded its behavior and observed failure modes.

In all cases, Paper2Agent first executes the tutorial/CLI examples to detect failures and only then attempts bounded, local repairs (e.g., fixing typos, updating deprecated calls, or installing missing dependencies) before re-running. For example, it repaired injected cell-level typos in AlphaGenome, corrected import/name typos in POP-TOOLS, and installed missing R packages in mlearner in our adversarial repositories. These fixes are applied to execution artifacts for Paper2Agent (not silently to the upstream repository) and logged. Across three independent runs of these adversarial cases, the end-to-end execution succeeded in 36/36 cases (100% success rate). Thus, our validation pipeline can surface and repair execution-level errors, including inherited bugs when present. However, if a repository contains bugs that do not trigger execution failures but instead produce systematically incorrect outputs, such errors are fundamentally difficult to detect automatically without external ground-truth references. We have added those new results in the Results and Discussion section of the revised paper:

Revised Results section:

“To further evaluate whether Paper2Agent inherits bugs from upstream codebases, we constructed adversarial variants of three representative repositories spanning different programming environments: AlphaGenome (Python-based), POP-TOOLS (Python command-line-based), and mlearner (R command-line-based). For each repository, we injected four categories of execution-level errors: (1) removal of required dependencies from environment specifications, (2) modification of input file paths to nonexistent locations, (3) insertion of typographical errors into code, and (4) replacement of API calls with deprecated equivalents, yielding 12 adversarial configurations in total. In each case, the agent’s iterative execution loop detected runtime failures from the error traceback, applied targeted local fixes, and re-executed the workflow. Across all 12 configurations with three independent runs, Paper2Agent completed end-to-end successfully (36/36), producing functional MCP servers with passing validation tests. All repairs were confined to the agent’s execution artifacts (e.g., patched notebook copies, wrapper scripts) rather than the upstream source, with the full repair history preserved in logs for traceability. These results suggest that Paper2Agent’s

execute-diagnose-repair loop provides a natural mechanism for detecting and correcting inherited bugs at execution time, rather than propagating them into the generated tools."

Revised Discussion:

"Not every paper can be seamlessly turned into a robust agent. As we demonstrate in the adversarial benchmarks, Paper2Agent's iterative execute-diagnose-repair loop can detect and correct many inherited bugs that manifest as execution failures. Nevertheless, in our large-scale evaluation, a substantial fraction of repositories still could not be successfully agentified, typically due to incomplete codebases, missing documentation, or unresolvable environment configurations. Moreover, bugs that do not trigger runtime exceptions but instead produce silently incorrect outputs, such as a flawed formula that returns plausible but wrong results, remain fundamentally difficult to detect without external ground-truth references, a limitation shared by any system that relies on execution-based validation. These challenges suggest that the ease with which a paper can be transformed into an agent may itself serve as a practical measure of reproducibility and rigor. Just as the scientific community has come to expect clear data and code availability, we envision a natural extension: expecting contributions to be structured in ways that facilitate their translation into agents. Well-documented, modular, and transparent papers will naturally lend themselves to this new standard."

Referee #2 (Remarks to the Author):

A. Summary of the key results

The article introduces Paper2Agent, an automated multi-agent pipeline that converts a research paper + public codebase into a runnable Model Context Protocol (MCP) server paired with a conversational agent. The system implements several components--including the environment-manager, tutorial-scanner, tutorial-tool-extractor-implementor, and test-verifier-improver--that produce MCP tools, resources, and prompts.

The authors evaluate Paper2Agent on three biological case studies: AlphaGenome (genomics), TISSUE (spatial transcriptomics), and Scanpy (single-cell analysis). For AlphaGenome, Paper2Agent automatically generated 22 MCP tools in ~3 hours and achieved 100% accuracy on 15 tutorial-based and 15 novel benchmark queries (manually graded vs. ground-truth code execution), while also reporting run-time speedups relative to Biomni and Claude + Code repo. Finally, the authors illustrate cross-agent integration (AlphaGenome + ADHD GWAS), where an AI co-scientist prioritized causal variants across 39 risk loci and proposed mechanistic hypotheses.

B. Originality and significance

The manuscript presents a novel system-level contribution: automated conversion of research papers into interactive MCP servers + conversational agents. While the broader idea of executable/reproducible research—e.g., executable papers, Jupyter notebooks, Papers With Code—has a long history, Paper2Agent's emphasis on automated tool extraction, structured MCP interfaces, and agent integration represents a meaningful extension of this paradigm.

Response: Thank you very much for your helpful feedback and for highlighting the value of Paper2Agent! We truly appreciate your review, which has really helped us to improve our paper.

In terms of technical novelty, the approach appears to build upon and combine several prior ideas, but remains incremental relative to existing platforms that expose runnable artifacts—such as CodeOcean (tech.cornell.edu/built/code-ocean/) and Binder (mybinder.org/). The manuscript would benefit from citing these tools explicitly and clarifying how Paper2Agent differs from them.

Response: We thank the reviewer for this suggestion. We now explicitly cite and discuss these systems in the revised manuscript. We clarified how Paper2Agent differs conceptually and technically from these platforms. Binder and CodeOcean primarily provide infrastructure for executing code in reproducible environments, but they still require users to understand the codebase, APIs, and workflow structure. In contrast, Paper2Agent converts papers and codebases into structured tool interfaces (via MCP) that are directly usable by AI agents through natural language. This shifts the user interaction paradigm from manual execution of research code to agent-mediated execution, reasoning, and workflow composition.

In addition, Paper2Agent introduces several capabilities that go beyond executable artifact hosting, including automated tool extraction from tutorials, iterative test-driven validation and repair of extracted tools, structured exposure of paper knowledge through resources and prompts, and seamless integration into multi-agent scientific workflows. We have clarified these distinctions in the revised manuscript to better position Paper2Agent relative to existing executable research infrastructure.

Specifically, we added the following text to the Introduction:

"Efforts to make research outputs more executable and accessible have been ongoing for years. Executable papers, such as those proposed in Elsevier's Executable Paper Grand Challenge and more recent Jupyter Notebook–backed publications, aimed to merge narrative text with runnable code. These approaches increased reproducibility but still required substantial technical familiarity to use fully. The Papers with Code initiative also aimed to bridge papers and implementations by linking publications to open source repositories. More recently, executable artifact platforms such as Binder and CodeOcean provide containerized environments for running code associated with papers, while paper-to-code systems like Paper2Code automatically generate functional code repositories from research papers. While these efforts improved reproducibility, there were still substantial barriers to understanding, customizing, and applying code to new projects."

In terms of significance, Paper2Agent has potentially high practical impact: reducing barriers to method adoption and enabling agentic workflows is highly valuable for computational biology and other computational science more broadly.

C. Data & methodology: validity of approach, quality of data, quality of presentation

The workflow used to convert a codebase into an MCP server (codebase extraction, environment configuration, tool synthesis, testing, and refinement) is reasonable. Using MCP as a standard interface for LLM integration is a sound design choice. The manuscript is generally well written, and the provided code repository enables reproducibility. For improved readability, the authors should include a clearer description of how the MCP servers are created end-to-end.

Response: We thank the reviewer for the positive assessment and for this helpful suggestion. In the revised manuscript, we added a clearer end-to-end description of MCP server creation to improve readability.

Specifically, we have revised our Extended Methods section to include a step-by-step overview of the full conversion pipeline, covering codebase extraction, environment reconstruction, tool synthesis, automated test generation, validation, and iterative refinement. We also clarified how these steps interact with the multi-agent orchestration framework and how outputs are compiled into a deployable MCP server. We added a concise summary of the inputs, intermediate

artifacts (e.g., extracted tool schemas, generated tests, execution logs), and final outputs produced during MCP construction. These revisions provide a clearer end-to-end view of MCP server creation and improve accessibility for readers seeking to understand or reproduce the system.

Revised Extended methods:

"Details on implementing Paper2Agent

Paper2Agent converts a research paper and its public codebase into a production-ready MCP server and then exposes that server to an AI agent interface. We implemented this as a multi-agent system using Claude Code's agent SDK, where a central orchestrator agent coordinates specialized sub-agents through a six-step pipeline. Each sub-agent is defined by a structured prompt that specifies its role, permitted tools (for example file read/write, shell execution, and web access), and expected output schema. The orchestrator dispatches sub-agents sequentially across steps and in parallel within steps when multiple tutorials are processed concurrently.

The pipeline proceeds through six steps, with data flowing between steps via standardized JSON reports and file conventions:

1. *Locate and download the codebase. Paper2Agent first attempts to automatically identify the associated code repository from the manuscript text, references, or supplementary materials. If automatic identification fails, if it returns multiple candidates, or if the user prefers to specify a particular repository, the repository URL can be provided directly. Once identified, the codebase is cloned or downloaded, along with associated resources such as supplementary data or configuration files.* ***The outputs for this step are the cloned repository and detected language.***

2. *Environment setup. The environment-manager sub-agent provides a clean, isolated virtual environment for the repository.* ***The input is the cloned repository, and the outputs are an isolated virtual environment and test configuration files.***

3. *Tutorial discovery. The tutorial-scanner sub-agent scans the repository to locate useful reference and educational materials and produce an index of candidate tutorials for tooling. The inputs are the cloned repository and an optional tutorial filter.* ***The output is a JSON file representing a classified file index.***

4. *Tutorial execution and audit. The tutorial-executor sub-agent runs the selected tutorials end-to-end with their example data, captures inputs, outputs, figures, and runtime constraints, and records any implicit assumptions that must be made explicit.* ***The inputs are tutorial source files, activated virtual environment, and scanner report. The outputs are executed notebooks and per-tutorial execution reports.***

5. *Tool extraction, testing, and refinement. This step involves two sub-agents operating in sequence. First, the tutorial-tool-extractor-implementor converts each executed tutorial into a standalone Python module containing reusable functions. It identifies generalizable analysis steps, parameterizes hardcoded values such as file paths, thresholds, and column names, enforces file-based inputs and outputs, and decorates each function as an MCP tool. Second, the test-verifier-improver creates per-function test files using the tutorial's own example data as ground truth. Tests verify that expected output files are generated, and functions that repeatedly fail have their MCP tool decorators removed and are excluded from the final server.* ***The inputs are executed notebooks, virtual environment, and scanner report. The outputs are tool modules, per-function test files, test logs, and summaries.***

6. *MCP server assembly. The orchestrator integrates all validated tool modules into a unified MCP server with a manifest, versioning, and basic security defaults, ready to be used by an orchestrator or co-scientist agent."*

Each sub-agent is instantiated as an independent LLM session (Claude) with a role-specific system prompt and a defined set of permitted tools (file read/write, shell execution, code search).

- ***Environment-manager****: a specialized agent responsible for creating clean, reproducible environments for research codebases. It analyzes project setup requirements, provisions an isolated workspace, installs all necessary dependencies, and ensures the code runs without conflicts. Standardizing environment setup enables reliable execution and reproducibility across different systems.*

- ***Tutorial-scanner****: a specialized agent for reviewing public codebases to identify and organize educational resources. It systematically scans available materials, distinguishes genuine tutorials from other files, and highlights those most useful for reuse. The agent then produces clear summaries and reports, providing a structured view of which resources are worth keeping and which can be set aside.*

- ***Tutorial-executor****: executes approved tutorials end-to-end to generate gold-standard outputs and reference data for downstream tool extraction. The agent systematically resolves execution errors, preserves all generated outputs such as numerical results, figures, and tables, and records execution metadata. The resulting executed notebooks, extracted figures, and generated data files serve as authoritative reference material for test creation and validation.*
- ***Tutorial-tool-extractor-implementor****: a specialized agent that converts tutorials into reusable tools. It reviews selected tutorials, identifies tasks that generalize beyond the example data, and implements each as a clean, single-purpose function with clear inputs, outputs, and defaults. The agent parameterizes hardcoded values, enforces file-based inputs, saves essential results and figures, and returns a standardized summary of produced artifacts. Its goal is to create a practical function library that*

reproduces tutorial results on the original data while remaining ready to run on new datasets.

- ***Test-verifier-improver**: a specialized agent that creates, runs, and refines tests for tutorial implementations. It uses only the tutorial's own examples to ensure complete coverage and faithful reproduction of numerical and visualization results. A test passes when expected files are generated, numerical results match tutorial outputs exactly with 3% tolerance for floating-point values, and generated figures match reference visualizations verified via perceptual hashing with Hamming distance less than 20. The agent runs in a loop of generating tests, executing them, diagnosing failures, and applying fixes, with a maximum of 6 attempts per function. If functions repeatedly fail, their MCP decorators are removed, a failure comment is added, and they are not included in the MCP server. All results and logs are recorded for transparency."*

D. Appropriate use of statistics and treatment of uncertainties

In the AlphaGenome benchmarking experiment, accuracy is reported without uncertainty estimates. Given that LLMs are stochastic, it would be useful to run the AlphaGenome Agent, Claude + repo, and Biomni multiple times to report variance or confidence intervals around performance.

Response: We thank the reviewer for this suggestion. We have added the analysis that ran each method five times [5 runs × (15 + 15) questions = 150 evaluations per method] and reported mean accuracy ± SE across runs.

Method	Tutorial tasks (mean ± SE)	Novel tasks (mean ± SE)
AlphaGenome Agent	98.7% ± 1.3%	100.0% ± 0.0%
Claude + repo	82.7% ± 3.4%	78.7% ± 4.4%
Biomni	37.3% ± 4.0%	56.0% ± 3.4%

To assess whether AlphaGenome agent significantly outperforms the baselines, we conducted one-sided paired t-tests on per-run accuracy. AlphaGenome agent significantly outperformed both Claude + repo (tutorial: $p = 8.1 \times 10^{-3}$; novel: $p = 4.2 \times 10^{-3}$) and Biomni (tutorial: $p = 4.7 \times 10^{-5}$; novel: $p = 1.0 \times 10^{-4}$). This is consistent with the original conclusion of the paper showing that AlphaGenome agent generated by Paper2Agent outperforms alternative approaches. For other newly added benchmark results, we have also added the mean ± SE for the reported accuracy.

We have added these results to the Results section of the revised manuscript.

Revised Method section:

"Across five independent runs, the AlphaGenome agent achieved 98.7% ± 1.3% accuracy on these queries, significantly higher than Claude + Repo (82.7% ± 3.4%, $p = 8.1 \times 10^{-3}$) and Biomni (37.3% ± 4.0%, $p = 4.7 \times 10^{-5}$)."

"Across five independent runs, the AlphaGenome agent achieved 100.0% ± 0.0% accuracy, as verified by manual execution of the original AlphaGenome code and evaluated by predefined rubrics. This is significantly higher than Claude + Repo (78.7% ± 4.4%, $p = 4.2 \times 10^{-3}$) and Biomni (56.0% ± 3.4%, $p = 1.0 \times 10^{-4}$)."

E. Conclusions: robustness, validity, reliability

The three biological case studies demonstrate that Paper2Agent can effectively automate runnable agents for well-documented research papers. While this is promising, the evidence for general robustness is limited — the evaluated papers have actively maintained codebases and are likely easier than the average research paper, which many lack comprehensive documentation. The authors do acknowledge these limitations in cases where code is incomplete or poorly documented.

Response: We thank the reviewer for this suggestion. Conceptually, Paper2Agent produces two complementary components for each paper: (i) a structured resource layer that exposes the main manuscript, supplementary tables, and metadata for retrieval and question answering, and (ii) an executable tool layer that exposes the computational methods described in the paper as callable MCP tools. For computational papers and other papers, the first component is always achievable, while the second depends on the quality and agentifiability of the associated code repository.

To quantify the second component, we conducted a systematic evaluation of 100 computational research papers randomly selected from bioRxiv's bioinformatics category. We define successful agentification in terms of execution capability: a paper was considered successfully agentified if the generated MCP server completed tool extraction, execution, and automated validation end-to-end without human intervention. We report the overall success rate, the fraction of tools attempted versus retained, and common failure modes, and analyze predictive factors such as documentation quality, dependency completeness, and tutorial availability.

Using this definition, 74 of 100 papers (74%) were successfully agentified with executable tools. The average cost for processing one successfully agentified paper is $14.99 and average time is 2.4 hours; papers that failed to agentify typically failed early in the pipeline (average cost $9.03, average time 1.7 hours). We evaluated these 74 paper agents on 300 benchmark questions derived from GitHub tutorials, achieving an accuracy of 91.2 ± 1.6% (mean ± bootstrap SE), significantly outperforming the baseline (Claude Code with direct repository access) at 80.3 ± 2.3% ($p < 0.0001$, bootstrap test; 95% CI of difference: [5.8%, 15.8%]). Paper2Agent answered benchmark questions at an average cost of $0.20 per query in 1.6

minutes, compared to the baseline's $0.38 per query in 4.3 minutes, indicating a 1.9x reduction in cost and 2.7x speedup in response time.

For the 26 papers that could not be fully agentified, we identified the following failure modes:

- No executable code (9 papers, 35%): repository was incomplete (e.g., "code will be released after acceptance") or contained only data files with no code.
- Missing data/model prerequisites (8 papers, 31%): required data files, model weights, or external resources not included in repository
- Environment/dependency failures (6 papers, 23%): unresolvable package conflicts or missing system libraries
- No extractable API (3 papers, 12%): the agent determined that the repository code was not generalizable for analyzing new data (e.g., reproducibility scripts hardcoded to specific datasets, GUI-only workflows)

These failure modes strongly correlate with documentation completeness, explicit environment specifications, and the presence of executable tutorials, which together explain the majority of agentification failures. In such cases, Paper2Agent's feedback can help the authors in making their code more usable.

We note that even when the executable tool layer cannot be constructed, Paper2Agent still provides value through its structured resource layer (manuscript text, supplementary tables, and metadata), enabling retrieval-augmented question answering for all papers. To benchmark this capability, we curated 100 questions from 13 bioRxiv preprints and 13 Nature articles. Paper2Agent achieved 89.0 ± 3.1% accuracy on tasks requiring synthesis of information across main text and supplementary materials, for example: "The paper reports results using Pearson correlation; reanalyze the conclusions using Spearman correlation." This significantly outperforms our baseline using Claude with browser-use capabilities and the paper URL, representing a strong human-assisted LLM setup, that achieved 82.0 ± 3.8% accuracy ($p = 0.03$, bootstrap test). Moreover, Paper2Agent was 34× cheaper ($0.27 vs $9.04 per question) and 15× faster (63s vs 919s per question).

Together, these evaluations demonstrate that Paper2Agent provides practical value across a broad spectrum of computational papers. For papers with well-maintained codebases, Paper2Agent enables fully automated tool construction with high accuracy on computational tasks. For the remaining papers and indeed for all papers regardless of code quality, the structured resource layer supports accurate information synthesis from main text and supplementary materials. These results establish both the scope and limitations of automated agentification: while executable tool generation depends on repository quality, the core capability of transforming papers into queryable, task-ready agents generalizes reliably across the scientific literature.

We have added those new results in our revised paper.

Revised Results section:

“Next, we conducted a large-scale evaluation across a heterogeneous corpus of research papers. We randomly selected 100 computational research papers from bioRxiv’s bioinformatics category, 26 general research papers from bioRxiv and Nature that focus on data and discovery rather than executable methods, and 5 computational papers from non-biology domains including AI and statistics. All papers were processed end-to-end by Paper2Agent without manual cleanup, code modification, or intervention.”

Paper2Agent produces two complementary components for each paper: (i) a structured resource layer that exposes the manuscript text, supplementary information, and metadata for retrieval and question answering, and (ii) an executable tool layer that exposes computational methods as callable MCP tools. While the resource layer is always constructible, successful tool-layer construction depends on the quality and agentifiability of the associated code repository.

“Among the 100 computational papers, 74 of 100 (74%) were successfully agentified with executable tools. The average cost for processing one successfully agentified paper was $14.99 with an average runtime of 2.4 hours; papers that failed typically failed early in the pipeline (average cost $9.03, average time 1.7 hours). We evaluated these 74 paper agents on 300 benchmark questions derived from GitHub tutorials. Paper2Agent achieved 91.2 ± 1.6% accuracy (mean ± bootstrap SE), significantly outperforming the baseline (Claude Code with direct repository access) at 80.3 ± 2.3% ($p < 0.0001$, bootstrap test; 95% CI of difference: [5.8%, 15.8%]). Paper2Agent answered benchmark questions at an average cost of $0.20 per query in 1.6 minutes, compared to $0.38 per query in 4.3 minutes for the baseline, corresponding to a 1.9x cost reduction and 2.7x speedup. For the remaining 26 computational papers that could not be fully agentified, dominant failure modes included missing executable code (35%), missing data or model artifacts (31%), environment or dependency failures (23%), and non-generalizable scripts (12%). These failures strongly correlate with documentation completeness, environment specification quality, and the presence of executable tutorials.

“Even when executable tools cannot be constructed, Paper2Agent still provides value through its structured resource layer. Across 26 non-computational papers (13 bioRxiv and 13 Nature papers), we curated 100 synthesis-based benchmark questions spanning main text and supplementary materials. Paper2Agent achieved 89.0 ± 3.1% accuracy, significantly outperforming a strong Claude browser-use baseline (82.0 ± 3.8%, $p = 0.03$), while being 34× cheaper and 15× faster. To understand whether the agent will reject out-of-scope queries, we constructed an adversarial out-of-scope benchmark by randomly permuting paper–question pairs across the 26 data and discovery papers. Paper2Agent achieved a 100% correct rejection rate under both prompted and unprompted conditions, reliably declining to answer mismatched paper–question pairs even without explicit instructions to do so.”

F. Suggested improvements: experiments, data for possible revision

1. Adaptation and training: In many cases, methods require tuning for specific use cases (e.g., hyperparameter adjustment). Can Paper2Agent handle such adaptation? Additionally, is it capable of training models, or is it limited to running pre-existing code?

Response: We thank the reviewer for this question. We have added two case studies demonstrating that Paper2Agent can perform model training from scratch and adaptively tune hyperparameters, rather than being limited to executing pre-existing code.

In the first case study (SAELens paper), Paper2Agent was tasked with training a Sparse Autoencoder (SAE) on the tiny-stories-1L-21M language model using the apollo-research/roneneldan-TinyStories-tokenizer-gpt2 dataset, with Weights & Biases for monitoring, and required to achieve a cross-entropy (CE) loss recovered of at least 0.95. The agent autonomously conducted three training iterations, progressively adjusting hyperparameters based on intermediate evaluation results. In the first iteration (L1 coefficient = 5, 1,000 training steps), the model achieved a CE Loss Recovered of 0.8529. The agent reduced the L1 coefficient to 2 and increased training steps to 5,000, achieving 0.9425. In the third iteration, it further reduced the L1 coefficient to 1 and increased training steps to 10,000, achieving a final CE Loss Recovered of 0.9736, exceeding the target threshold.

In the second case study (TabPFN paper), Paper2Agent trained and compared multiple models on benchmark datasets. On the breast cancer Wisconsin dataset, the agent performed 5-fold cross-validation and found that TabPFN achieved the highest ROC AUC (0.9968), outperforming CatBoost (0.9947), XGBoost (0.9935), and Random Forest (0.9885). On the California Housing dataset, TabPFN also achieved the best performance with an R^2 of 0.8708, compared to CatBoost (0.8498), XGBoost (0.8313), and Random Forest (0.8061).

We have added those results in the revised paper:

Revised Results section:

"We further extend our evaluation to include five non-biology case studies including diverse scientific domains, spanning causal inference (grf), mechanistic interpretability (SAELens), natural language processing (Binoculars), computer vision (SAM2), and tabular machine learning (TabPFN). Across 17 execution-based benchmark tasks, Paper2Agent achieved 95.3 ± 1.9% accuracy across five independent runs, demonstrating generalization beyond computational biology. Among those tasks, we also examined whether Paper2Agent can handle model training and iterative hyperparameter adaptation. In the SAELens case study, the agent trained a Sparse Autoencoder on a small language model, and upon failing to meet the target reconstruction loss, autonomously adjusted hyperparameters across three iterations until exceeding the threshold. In the TabPFN case study, the agent independently trained and compared four models using cross-validation on benchmark datasets. These results demonstrate that Paper2Agent is not limited to running pre-existing code but can also perform model training and adaptive hyperparameter tuning."

2. Ablation studies: The workflow includes multiple sub-agents performing distinct tasks. An ablation study—e.g., disabling the test-verifier or tutorial-scanning components—would clarify the contribution of each component to overall performance.

Response: We thank the reviewer for this suggestion. The multi-agent design was originally chosen for three main reasons: (i) to prevent context-window saturation during long-horizon code analysis (we observed that collapsing all steps into a single agent leads to severe context blowup and degraded tool quality), (ii) to enable parallel execution of independent sub-tasks, including environment setup, tutorial discovery, tool extraction, and verification, (iii) to ensure the implemented MCP Python script is correct through the test-verifier-improver.

To evaluate whether this modular design is necessary, we conducted an ablation study using the AlphaGenome repository with three simplified variants of the pipeline. Each condition was run three times to estimate stability and success rates: (1) Monolithic agent: all sub-tasks are executed sequentially within a single agent and a single context window (200K tokens) (2) Non-parallel multi-agent: the same sub-agents are used, but forced to run sequentially rather than in parallel. (3) No test-verifier-improver agent.

Compared to the full multi-agent pipeline, the monolithic agent failed to generate a valid MCP server Python script in all three runs. Failure analysis revealed that the agent was unable to maintain coherent long-horizon execution: in run 1, the agent prematurely requested to terminate the workflow ("Would you like me to continue with Phase 3?"); in run 2, it recommended skipping critical steps entirely ("Tool extraction is not feasible"); and in run 3, it explicitly admitted failure ("NOT COMPLETED… due to scope and complexity"), achieving only partial completion (~40%).

In contrast, the non-parallel multi-agent variant was able to generate a usable MCP server across all three runs, but required substantially longer wall-clock time. Specifically, this variant took 2.2–2.4x longer (93m13s – 104m44s per run) compared to the parallel multi-agent system (42m23s – 43m39s per run). Together, these results indicate that both modularization and parallel execution are important for reliable and efficient agentification. We have revised the results section to add these new results.

Finally, removing the test-verifier-improver agent led to unstable tool quality: although MCP servers were still generated, several extracted tools failed to reproduce tutorial outputs or produced silent numerical errors that were detectable only through manual inspection. On our tutorial and novel benchmarks, accuracy dropped to 69.3 ± 4.5% and 84.0 ± 2.7% respectively, compared to 98.7% ± 1.3% and 100.0% ± 0.0% with the full system, demonstrating that automated verification is essential for ensuring tool correctness, not merely a convenience. We have added those new results to our revised manuscript:

Revised Results section:

"Next, we conducted ablation studies to evaluate the benefit of Paper2Agent's architectural design choices, including the multi-agent orchestration, parallel execution, automated

test-driven verification, the MCP tool interface, and agent scaffolding. We performed these studies on the AlphaGenome repository by comparing the full Paper2Agent system with several simplified variants: (1) a monolithic agent, where all sub-tasks are executed within a single context window; (2) a non-parallel multi-agent system, where sub-agents run sequentially; (3) a system without the test-verifier-improver agent; (4) an alternative implementation using Markdown skill files instead of MCP tools; and (5) an alternative coding agent (OpenCode) to test scaffolding robustness.

The monolithic agent failed to generate a valid MCP server in all three runs due to context management and planning failures. The non-parallel multi-agent variant succeeded but required 2.2–2.4× longer runtime. Removing the test-verifier-improver agent led to unstable tool quality: accuracy dropped to 69.3 ± 4.5% and 84.0 ± 2.7% on tutorial and novel benchmarks, respectively, compared to 98.7% ± 1.3% and 100.0% ± 0.0% with the full system."

3. Failure modes and robustness: Quantify failure rates, such as when MCP fails to generate tools or when tests fail and are pruned. should be quantified.

Response: We thank the reviewer for this suggestion. We have added the analysis to quantify failure rates across our large-scale evaluation of 100 computational papers. Of these, 26 papers (26%) could not be fully agentified, with dominant failure modes including missing executable code (35%), missing data or model artifacts (31%), environment or dependency failures (23%), and non-generalizable scripts (12%). We further tested Paper2Agent on 12 adversarial repository variants with injected errors (missing dependencies, incorrect paths, code typos, deprecated API calls) across three repositories spanning Python, command-line, and R environments. Paper2Agent successfully diagnosed and repaired all 12 configurations (100% recovery rate) across three independent runs. These results are now reported in the revised Results section.

Revised Results section:

"For the remaining 26 computational papers that could not be fully agentified, dominant failure modes included missing executable code (35%), missing data or model artifacts (31%), environment or dependency failures (23%), and non-generalizable scripts (12%). These failures strongly correlate with documentation completeness, environment specification quality, and the presence of executable tutorials."

"We also conducted adversarial stress tests of Paper2Agent to assess robustness to repository defects and out-of-scope query scenarios. To test robustness to upstream code failures, we constructed adversarial variants of three representative repositories spanning different programming environments: AlphaGenome (Python-based), POP-TOOLS (Python command-line-based), and mlearner (R command-line-based). For each repository, we injected four categories of execution-level errors: (1) removal of required dependencies from environment specifications, (2) modification of input file paths to nonexistent locations, (3) insertion of typographical errors into code, and (4) replacement of API calls with deprecated equivalents, yielding 12 adversarial configurations in total. In each case, the agent's iterative

execution loop detected runtime failures from error tracebacks, applied targeted local fixes, and re-executed the workflow. Across all 12 configurations with three independent runs, Paper2Agent completed end-to-end successfully (36/36), producing functional MCP servers with passing validation tests."

4. Hallucination: Has the hallucination rate been evaluated? One approach is to introduce adversarial inputs, such as modified READMEs or misleading tutorials.

Response: We thank the reviewer for this suggestion. We evaluated hallucination behavior through an out-of-scope query benchmark. Specifically, we constructed adversarial paper–question pairs by randomly permuting questions across 26 non-computational papers, ensuring each question was paired with a paper that does not contain the relevant information. Under instructions to answer only using local paper files, Paper2Agent achieved a 100% correct rejection rate, consistently declining to answer mismatched questions rather than hallucinating plausible but unsupported answers. In addition, we tested Paper2Agent on 12 adversarial repository variants with injected errors across three repositories. In all cases with three independent runs, the agent detected failures through execution rather than silently producing incorrect outputs, and successfully repaired all 12 configurations. Together, these evaluations demonstrate that Paper2Agent is robust to both out-of-scope queries and adversarial codebase conditions. These results are now reported in the revised Results section.

Revised Results section:

"We also conducted adversarial stress tests of Paper2Agent to assess robustness to repository defects and out-of-scope query scenarios. To test robustness to upstream code failures, we constructed adversarial variants of three representative repositories spanning different programming environments: AlphaGenome (Python-based), POP-TOOLS (Python command-line-based), and mlearner (R command-line-based). For each repository, we injected four categories of execution-level errors: (1) removal of required dependencies from environment specifications, (2) modification of input file paths to nonexistent locations, (3) insertion of typographical errors into code, and (4) replacement of API calls with deprecated equivalents, yielding 12 adversarial configurations in total. In each case, the agent's iterative execution loop detected runtime failures from error tracebacks, applied targeted local fixes, and re-executed the workflow. Across all 12 configurations with three independent runs, Paper2Agent completed end-to-end successfully (36/36), producing functional MCP servers with passing validation tests."

5. Performance on incomplete or poorly documented repositories: The authors acknowledge that a limitation of Paper2Agent arises when the code repository is not well documented. The examples in the manuscript all use polished, well-maintained repositories, but in practice, many papers have incomplete or poorly maintained code. Therefore, the authors should evaluate Paper2Agent on such repositories to better reflect real-world scenarios. One approach could be to intentionally hide parts of the repositories used in the article and observe the agent's response. An even stronger demonstration would be to include case studies that more closely reflect real-world settings.

Response: We thank the reviewer for this suggestion. We have added controlled adversarial experiments with injected defect to understand Paper2Agent's response. We constructed adversarial variants of three representative repositories spanning different programming paradigms: AlphaGenome (Python-based), POP-TOOLS (Python command-line-based), and mlearner (R-based). For each repository, we injected four categories of execution-level errors that simulate common real-world defects: (1) removal of required dependencies from environment specifications (e.g., deleting entries from requirements.txt), (2) modification of input file paths to nonexistent locations, (3) insertion of typographical errors into code cells or scripts, and (4) replacement of API calls with deprecated equivalents. This yielded 12 adversarial configurations in total. In each case, Paper2Agent's iterative execute-diagnose-repair loop detected runtime failures from error tracebacks, applied targeted local fixes (e.g., installing missing packages, correcting import paths, updating deprecated function calls), and re-executed the workflow. Across all 12 configurations with three independent runs, Paper2Agent completed end-to-end successfully (36/36, 100% success rate), producing functional MCP servers with passing validation tests. We have added those new results in our revised paper.

Revised Results section:

"We also conducted adversarial stress tests of Paper2Agent to assess robustness to repository defects and out-of-scope query scenarios. To test robustness to upstream code failures, we constructed adversarial variants of three representative repositories spanning different programming environments: AlphaGenome (Python-based), POP-TOOLS (Python command-line-based), and mlearner (R command-line-based). For each repository, we injected four categories of execution-level errors: (1) removal of required dependencies from environment specifications, (2) modification of input file paths to nonexistent locations, (3) insertion of typographical errors into code, and (4) replacement of API calls with deprecated equivalents, yielding 12 adversarial configurations in total. In each case, the agent's iterative execution loop detected runtime failures from error tracebacks, applied targeted local fixes, and re-executed the workflow. Across all 12 configurations with three independent runs, Paper2Agent completed end-to-end successfully (36/36), producing functional MCP servers with passing validation tests."

6. Statistical evaluation: Since LLMs are stochastic, running AlphaGenome Agent, Claude + repo, and Biomni multiple times to report variance or confidence intervals would strengthen the analysis.

Response: We thank the reviewer for this suggestion. We have added the analysis that ran each method five times [5 runs × (15 + 15) questions = 150 evaluations per method] and reported mean accuracy ± SE across runs.

Method	Tutorial tasks (mean ± SE)	Novel tasks (mean ± SE)

AlphaGenome Agent	98.7% ± 1.3%	100.0% ± 0.0%
Claude + repo	82.7% ± 3.4%	78.7% ± 4.4%
Biomni	37.3% ± 4.0%	56.0% ± 3.4%

To assess whether AlphaGenome agent significantly outperforms the baselines, we conducted one-sided paired t-tests on per-run accuracy. AlphaGenome agent significantly outperformed both Claude + repo (tutorial: $p = 8.1 \times 10^{-3}$; novel: $p = 4.2 \times 10^{-3}$) and Biomni (tutorial: $p = 4.7 \times 10^{-5}$; novel: $p = 1.0 \times 10^{-4}$). This is consistent with the original conclusion of the paper showing that AlphaGenome agent generated by Paper2Agent outperforms alternative approaches. For other newly added benchmark results, we have also added the mean ± SE for the reported accuracy.

We have added these results to the Results section of the revised manuscript.

Revised Method section:

"Across five independent runs, the AlphaGenome agent achieved 98.7% ± 1.3% accuracy on these queries, significantly higher than Claude + Repo (82.7% ± 3.4%, $p = 8.1 \times 10^{-3}$) and Biomni (37.3% ± 4.0%, $p = 4.7 \times 10^{-5}$)."

"Across five independent runs, the AlphaGenome agent achieved 100.0% ± 0.0% accuracy, as verified by manual execution of the original AlphaGenome code and evaluated by predefined rubrics. This is significantly higher than Claude + Repo (78.7% ± 4.4%, $p = 4.2 \times 10^{-3}$) and Biomni (56.0% ± 3.4%, $p = 1.0 \times 10^{-4}$)."

7. Readability and clarity: The manuscript should explicitly list the contributions of Paper2Agent. In the extended methods, the sub-agents created for their workflow are described, but their creation process and corresponding inputs/outputs are unclear. Details on constructing the MCP servers are also missing and should be included.

Response: We thank the reviewer for this helpful suggestion. We have revised the manuscript to explicitly list the key contributions of Paper2Agent in the Introduction and to improve clarity around the system components. In the Extended Methods, we now provide a clearer description of each sub-agent, including its role, inputs, and outputs, as well as the data flow between agents. We have also expanded the description of the MCP server construction process, detailing how tools, resources, and prompts are generated, validated, and assembled. These revisions make the workflow and system architecture more transparent and easier to follow.

Revised Extended methods:

"Details on implementing Paper2Agent

Paper2Agent converts a research paper and its public codebase into a production-ready MCP server and then exposes that server to an AI agent interface. We implemented this as a multi-agent system using Claude Code's agent SDK, where a central orchestrator agent coordinates specialized sub-agents through a six-step pipeline. Each sub-agent is defined by a structured prompt that specifies its role, permitted tools (for example file read/write, shell execution, and web access), and expected output schema. The orchestrator dispatches sub-agents sequentially across steps and in parallel within steps when multiple tutorials are processed concurrently.

The pipeline proceeds through six steps, with data flowing between steps via standardized JSON reports and file conventions:

7. *Locate and download the codebase. Paper2Agent first attempts to automatically identify the associated code repository from the manuscript text, references, or supplementary materials. If automatic identification fails, if it returns multiple candidates, or if the user prefers to specify a particular repository, the repository URL can be provided directly. Once identified, the codebase is cloned or downloaded, along with associated resources such as supplementary data or configuration files.* ***The outputs for this step are the cloned repository and detected language.***

8. *Environment setup. The environment-manager sub-agent provides a clean, isolated virtual environment for the repository.* ***The input is the cloned repository, and the outputs are an isolated virtual environment and test configuration files.***

9. *Tutorial discovery. The tutorial-scanner sub-agent scans the repository to locate useful reference and educational materials and produce an index of candidate tutorials for tooling. The inputs are the cloned repository and an optional tutorial filter.* ***The output is a JSON file representing a classified file index.***

10. *Tutorial execution and audit. The tutorial-executor sub-agent runs the selected tutorials end-to-end with their example data, captures inputs, outputs, figures, and runtime constraints, and records any implicit assumptions that must be made explicit.* ***The inputs are tutorial source files, activated virtual environment, and scanner report. The outputs are executed notebooks and per-tutorial execution reports.***

11. *Tool extraction, testing, and refinement. This step involves two sub-agents operating in sequence. First, the tutorial-tool-extractor-implementor converts each executed tutorial into a standalone Python module containing reusable functions. It identifies generalizable analysis steps, parameterizes hardcoded values such as file paths, thresholds, and column names, enforces file-based inputs and outputs, and decorates each function as an MCP tool. Second, the test-verifier-improver creates per-function test files using the tutorial's own example data as ground truth. Tests verify that expected output files are generated, and functions that repeatedly fail have their MCP tool*

decorators removed and are excluded from the final server. ***The inputs are executed notebooks, virtual environment, and scanner report. The outputs are tool modules, per-function test files, test logs, and summaries.***

12. *MCP server assembly. The orchestrator integrates all validated tool modules into a unified MCP server with a manifest, versioning, and basic security defaults, ready to be used by an orchestrator or co-scientist agent."*

Each sub-agent is instantiated as an independent LLM session (Claude) with a role-specific system prompt and a defined set of permitted tools (file read/write, shell execution, code search).

- ***Environment-manager****: a specialized agent responsible for creating clean, reproducible environments for research codebases. It analyzes project setup requirements, provisions an isolated workspace, installs all necessary dependencies, and ensures the code runs without conflicts. Standardizing environment setup enables reliable execution and reproducibility across different systems.*

- ***Tutorial-scanner****: a specialized agent for reviewing public codebases to identify and organize educational resources. It systematically scans available materials, distinguishes genuine tutorials from other files, and highlights those most useful for reuse. The agent then produces clear summaries and reports, providing a structured view of which resources are worth keeping and which can be set aside.*

- ***Tutorial-executor****: executes approved tutorials end-to-end to generate gold-standard outputs and reference data for downstream tool extraction. The agent systematically resolves execution errors, preserves all generated outputs such as numerical results, figures, and tables, and records execution metadata. The resulting executed notebooks, extracted figures, and generated data files serve as authoritative reference material for test creation and validation.*
- ***Tutorial-tool-extractor-implementor****: a specialized agent that converts tutorials into reusable tools. It reviews selected tutorials, identifies tasks that generalize beyond the example data, and implements each as a clean, single-purpose function with clear inputs, outputs, and defaults. The agent parameterizes hardcoded values, enforces file-based inputs, saves essential results and figures, and returns a standardized summary of produced artifacts. Its goal is to create a practical function library that reproduces tutorial results on the original data while remaining ready to run on new datasets.*

- ***Test-verifier-improver****: a specialized agent that creates, runs, and refines tests for tutorial implementations. It uses only the tutorial's own examples to ensure complete coverage and faithful reproduction of numerical and visualization results. A test passes when expected files are generated, numerical results match tutorial outputs exactly with 3% tolerance for floating-point values, and generated figures match reference*

visualizations verified via perceptual hashing with Hamming distance less than 20. The agent runs in a loop of generating tests, executing them, diagnosing failures, and applying fixes, with a maximum of 6 attempts per function. If functions repeatedly fail, their MCP decorators are removed, a failure comment is added, and they are not included in the MCP server. All results and logs are recorded for transparency."

8. Robustness evaluation: It will be important to include more challenging, real-world case studies — e.g., scenarios with missing code and incomplete documentation.

Response: We thank the reviewer for this suggestion. To evaluate robustness under realistic, degraded repository conditions, we pursued two complementary strategies: (i) large-scale evaluation on naturally heterogeneous repositories, and (ii) controlled adversarial experiments with injected defects.

First, our large-scale evaluation of 100 randomly selected computational papers from bioRxiv's bioinformatics category inherently includes repositories with varying levels of documentation quality, code completeness, and maintenance status. These were processed end-to-end by Paper2Agent without manual cleanup or intervention. Of these, 74 papers (74%) were successfully agentified, and the 26 failures were primarily attributable to missing executable code (35%), missing data or model artifacts (31%), environment or dependency failures (23%), and non-generalizable scripts (12%). These results demonstrate that Paper2Agent is not limited to polished, well-maintained repositories and can handle the heterogeneity typical of real-world research code.

Second, to systematically assess robustness under controlled conditions, we constructed adversarial variants of three representative repositories spanning different programming paradigms: AlphaGenome (Python-based), POP-TOOLS (Python command-line-based), and mlearner (R-based). For each repository, we injected four categories of execution-level errors that simulate common real-world defects: (1) removal of required dependencies from environment specifications (e.g., deleting entries from requirements.txt), (2) modification of input file paths to nonexistent locations, (3) insertion of typographical errors into code cells or scripts, and (4) replacement of API calls with deprecated equivalents. This yielded 12 adversarial configurations in total. In each case, Paper2Agent's iterative execute-diagnose-repair loop detected runtime failures from error tracebacks, applied targeted local fixes (e.g., installing missing packages, correcting import paths, updating deprecated function calls), and re-executed the workflow. Across all 12 configurations with three independent runs, Paper2Agent completed end-to-end successfully (36/36, 100% success rate), producing functional MCP servers with passing validation tests.

Together, these evaluations demonstrate that Paper2Agent is robust to both the natural heterogeneity of real-world research repositories and to deliberately injected code defects, addressing concerns about performance being limited to well-maintained codebases. We have added these results to the revised manuscript.

G. References: appropriate credit to previous work?

Overall, the paper cites relevant prior work, and the experimental comparisons against Claude + Code and Biomni are appropriate. As noted earlier, the authors should clarify how Paper2Agent differs from existing platforms that convert code repositories into executable artifactors, and include those systems explicitly in the related work discussion.

Response: We thank the reviewer for this suggestion. In the revised manuscript, we have expanded the introduction section to explicitly discuss executable artifact platforms such as Binder and CodeOcean, as well as paper-to-code systems like Paper2Code, which automatically transform research papers into functional code repositories. We clarify that while these prior systems focus on generating runnable code from papers or providing executable environments for code associated with publications, Paper2Agent differs in its emphasis on automatically converting papers into interactive AI agents with composable tools. This enables research papers to be accessed as interactive AI agents through natural language, rather than merely providing runnable environments or static code artifacts. We have revised the introduction to add those previous works:

Revised introduction section:

"Efforts to make research outputs more executable and accessible have been ongoing for years. Executable papers, such as those proposed in Elsevier's Executable Paper Grand Challenge and more recent Jupyter Notebook–backed publications, aimed to merge narrative text with runnable code. These approaches increased reproducibility but still required substantial technical familiarity to use fully. The Papers with Code initiative also aimed to bridge papers and implementations by linking publications to open source repositories. More recently, executable artifact platforms such as Binder and CodeOcean provide containerized environments for running code associated with papers, while paper-to-code systems like Paper2Code automatically generate functional code repositories from research papers. While these efforts improved reproducibility, substantial barriers remain in understanding, customizing, and applying code to new projects."

H. Clarity and context: lucidity of abstract/summary, appropriateness of abstract, introduction and conclusions

The problem statement is clear and well motivated. The proposed solution is intuitive and presented in a reasonably accessible manner, and the use cases appear practically meaningful. The conclusion highlights the strengths of Paper2Agent and also acknowledge the limitations and future directions.

Minor comments

Figure 1B suggests that the codebase is identified from the paper, whereas the extended methods section states that the codebase is provided as part of the input. Please clarify what is expected as input to Paper2Agent.

Response: We thank the reviewer for pointing out this ambiguity. In our current implementation, Paper2Agent takes a research paper (PDF or URL) as input and first attempts to automatically identify the associated code repository from the manuscript text, references, or supplementary materials. If automatic identification fails, returns multiple candidates, or if the user prefers to specify a particular repository (e.g., to use only the tools from a specific codebase), the repository URL can be provided directly. We have revised the extended method section to clarify that codebase identification can be either automatic or user-specified:

Revised extended method section:

"Paper2Agent first attempts to automatically identify the associated code repository from the manuscript text, references, or supplementary materials. If automatic identification fails, returns multiple candidates, or if the user prefers to specify a particular repository, the repository URL can be provided directly. Once identified, the codebase is cloned or downloaded, along with associated resources such as supplementary data or configuration files."

The subplots in Figure 4C are difficult to read, particularly the cell type annotations. Increasing the font size or improving contrast would make the figure more interpretable.

Response: We thank the reviewer for the suggestion. We have increased the font size in Figure 4C to enhance readability and interpretability.

Referee #2 (Remarks on code availability):

The code is well-documented and includes multiple READMEs to support replication of the analyses in the manuscript. The instructions cover both basic and advanced usage scenarios (e.g., target tutorial processing and repositories which need authentication). In addition, the MCP servers for the three showcased code repositories have been publicly released on Hugging Face, further increasing usability.

Referee #3 (Remarks to the Author):

The manuscript introduces Paper2Agent, a systems framework that converts a research paper together with its public codebase into an MCP-based AI agent. The framework constructs an MCP server exposing executable tools, static resources, and workflow prompts derived from the paper and its tutorials, and connects this server to an LLM-based agent for natural-language interaction. The approach is demonstrated on three computational biology codebases with an additional illustrative application combining AlphaGenome with ADHD GWAS fine-mapping data.

The paper presents a clear and coherent systems abstraction for agentifying computational methods via MCP. The engineering pipeline is carefully designed, and the chosen case studies are relevant to computational biology and demonstrate that for well-maintained repositories, the framework can reproduce tutorial outputs and enable interactive use. This is a thoughtful and practically motivated systems paper that addresses a real pain point in method reuse and reproducibility. However, the empirical evaluation is narrow, with quantitative benchmarking limited to a single method (AlphaGenome) and a small, curated set of queries. As a result, claims regarding reliability, reproducibility, and generality are not fully supported. In addition, statistical quantification and analysis is minimal, and robustness to prompt variation, model choice, or run-to-run stochasticity is not evaluated.

Response: Thank you very much for your helpful suggestions! We really appreciate it. We have carefully revised our manuscript to incorporate your comments.

Major points

1. The contribution is mainly a systems and engineering integration, rather than a new algorithmic or modeling advance. The idea of enabling interactive execution of scientific methods builds on existing directions including executable papers, paper-to-code systems, and scientific agents. The main added value here is the combination of MCP-based packaging, automated tool extraction from tutorials, and test-verified execution, which is a great advance. But it would be important to clarify the novelty relative to existing work (e.g., executable papers, Paper2Code/PaperCoder, Biomni, CellVoyager). A comparison table identifying differences (e.g., tool synthesis vs. repo generation, multi-paper composition, etc) could help clarify what Paper2Agent enables that prior systems do not.

Response: We thank the reviewer for this helpful suggestion. Paper2Agent represents a paradigm shift from prior approaches to computational reproducibility and paper operationalization. Unlike environment-focused systems (Binder, CodeOcean) that require existing codebases and manual configuration, or code generation systems (Paper2Code) that output repositories for human developers, Paper2Agent automatically extracts computational methods from papers and packages them as reusable MCP servers with structured tool interfaces, resource access (full paper text and supplementary materials), and workflow prompts. This differs fundamentally from autonomous analysis agents (Biomni, CellVoyager) that execute domain-specific tasks on user queries: Paper2Agent generates reusable agent

infrastructure that any LLM or AI system can invoke via the standardized Model Context Protocol. Key unique capabilities include automated tool extraction from papers and code, multi-paper agent composition (enabling complex workflows that combine methods from multiple publications), cross-domain generalization (not limited to specific fields like biomedicine or single-cell analysis), and a two-phase design where agent generation is separated from agent execution. Importantly, Paper2Agent empowers users to design their own custom agents by selecting and combining methods from multiple papers based on their specific research needs, rather than being constrained to pre-built, domain-specific agents or manual code assembly. This architecture enables Paper2Agent to bridge the gap between static scientific literature and dynamic AI systems, making published research directly accessible as composable, LLM-callable tools rather than merely reproducible code or one-off analyses.

In the revised manuscript, we have added a comparative feature table positioning Paper2Agent relative to existing executable-artifact platforms and paper-to-code systems. This table is presented below and has been added to the Supplementary Table.

Feature	Binder	CodeOcean	Paper2Code	Biomni	CellVoyager	Paper2Agent
Input	Paper + code	Paper + code	Paper only	User query	Dataset + paper summary	Paper + code for agent generation; user queries for agent execution
Primary output	Executable environment	Executable environment	Generated code repo	Code-based analysis	Jupyter notebook	MCP server (tools + resources + prompts)
Requires existing codebase	Yes	Yes	No	No	No	Yes (when available)
Automated tool extraction	No	No	No	No	No	Yes
Structured tool interface (API schema)	No	No	No	No	No	Yes (MCP)
Natural language interaction	No	No	No	Yes	Yes	Yes
Multi-paper agent composition	No	No	No	No	No	Yes
Cross-domain generalization	N/A	N/A	ML papers	Biomedicine	scRNA-seq	Any domain
Reusable by other agents/LLMs	No	No	No	No	No	Yes (via MCP protocol)
User needs to understand code	Yes	Yes	Yes	No	No	No
Automated env setup & repair	Manual (Dockerfile)	Managed	No	Pre-installed env	Manual	Yes
Paper knowledge as resource	No	No	No	Partial (retrieval)	Paper summary	Yes (full text + suppl.)
Workflow templates (prompts)	No	No	No	No	No	Yes

2. The multi-agent pipeline (environment setup, tutorial scanning, tool extraction, testing and refinement) is reasonable. However, the evaluation conflates feasibility demonstrations with claims of general reliability. Only AlphaGenome is evaluated quantitatively, while Scanpy and TISSUE are assessed qualitatively via consistency with human-executed workflows. The ADHD GWAS example is best interpreted as an illustrative demonstration rather than a benchmark. The authors could add a small but systematic benchmark across a more diverse set of repositories (e.g., Python-only, R-based, mixed-language, limited documentation), reporting success/failure rates and failure modes.

Response: We thank the reviewer for this suggestion. Conceptually, Paper2Agent produces two complementary components for each paper: (i) a structured resource layer that exposes the main manuscript, supplementary tables, and metadata for retrieval and question answering, and (ii) an executable tool layer that exposes the computational methods described in the paper as callable MCP tools. For computational papers and other papers, the first component is always achievable, while the second depends on the quality and agentifiability of the associated code repository.

To quantify the second component, we conducted a systematic evaluation of 100 computational research papers randomly selected from bioRxiv's bioinformatics category. We define successful agentification in terms of execution capability: a paper was considered successfully agentified if the generated MCP server completed tool extraction, execution, and automated validation end-to-end without human intervention. We report the overall success rate, the fraction of tools attempted versus retained, and common failure modes, and analyze predictive factors such as documentation quality, dependency completeness, and tutorial availability.

Using this definition, 74 of 100 papers (74%) were successfully agentified with executable tools. The average cost for processing one successfully agentified paper is $14.99 and average time is 2.4 hours; papers that failed to agentify typically failed early in the pipeline (average cost $9.03, average time 1.7 hours). We evaluated these 74 paper agents on 300 benchmark questions derived from GitHub tutorials, achieving an accuracy of 91.2 ± 1.6% (mean ± bootstrap SE), significantly outperforming the baseline (Claude Code with direct repository access) at 80.3 ± 2.3% ($p < 0.0001$, bootstrap test; 95% CI of difference: [5.8%, 15.8%]). Paper2Agent answered benchmark questions at an average cost of $0.20 per query in 1.6 minutes, compared to the baseline's $0.38 per query in 4.3 minutes, indicating a 1.9x reduction in cost and 2.7x speedup in response time.

For the 26 papers that could not be fully agentified, we identified the following failure modes:

- No executable code (9 papers, 35%): repository was incomplete (e.g., "code will be released after acceptance") or contained only data files with no code.
- Missing data/model prerequisites (8 papers, 31%): required data files, model weights, or external resources not included in repository
- Environment/dependency failures (6 papers, 23%): unresolvable package conflicts or missing system libraries

- No extractable API (3 papers, 12%): the agent determined that the repository code was not generalizable for analyzing new data (e.g., reproducibility scripts hardcoded to specific datasets, GUI-only workflows)

These failure modes strongly correlate with documentation completeness, explicit environment specifications, and the presence of executable tutorials, which together explain the majority of agentification failures.

We note that even when the executable tool layer cannot be constructed, Paper2Agent still provides value through its structured resource layer (manuscript text, supplementary tables, and metadata), enabling retrieval-augmented question answering for all papers. To benchmark this capability, we curated 100 questions from 13 bioRxiv preprints and 13 Nature articles. Paper2Agent achieved 89.0 ± 3.1% accuracy on tasks requiring synthesis of information across main text and supplementary materials, for example: “The paper reports results using Pearson correlation; reanalyze the conclusions using Spearman correlation.” This significantly outperforms our baseline using Claude with browser-use capabilities and the paper URL, representing a strong human-assisted LLM setup, that achieved 82.0 ± 3.8% accuracy ($p = 0.03$, bootstrap test). Moreover, Paper2Agent was 34× cheaper ($0.27 vs $9.04 per question) and 15× faster (63s vs 919s per question).

Together, these evaluations demonstrate that Paper2Agent provides practical value across a broad spectrum of computational papers. For papers with well-maintained codebases, Paper2Agent enables fully automated tool construction with high accuracy on computational tasks. For the remaining papers and indeed for all papers regardless of code quality, the structured resource layer supports accurate information synthesis from main text and supplementary materials. These results establish both the scope and limitations of automated agentification: while executable tool generation depends on repository quality, the core capability of transforming papers into queryable, task-ready agents generalizes reliably across the scientific literature.

We have added those new results in our revised paper.

Revised Results section:

“Next, we conducted a large-scale evaluation across a heterogeneous corpus of research papers. We randomly selected 100 computational research papers from bioRxiv’s bioinformatics category, 26 general research papers from bioRxiv and Nature that focus on data and discovery rather than executable methods, and 5 computational papers from non-biology domains including AI and statistics. All papers were processed end-to-end by Paper2Agent without manual cleanup, code modification, or intervention.”

Paper2Agent produces two complementary components for each paper: (i) a structured resource layer that exposes the manuscript text, supplementary information, and metadata for retrieval and question answering, and (ii) an executable tool layer that exposes computational

methods as callable MCP tools. While the resource layer is always constructible, successful tool-layer construction depends on the quality and agentifiability of the associated code repository.

"Among the 100 computational papers, 74 of 100 (74%) were successfully agentified with executable tools. The average cost for processing one successfully agentified paper was $14.99 with an average runtime of 2.4 hours; papers that failed typically failed early in the pipeline (average cost $9.03, average time 1.7 hours). We evaluated these 74 paper agents on 300 benchmark questions derived from GitHub tutorials. Paper2Agent achieved 91.2 ± 1.6% accuracy (mean ± bootstrap SE), significantly outperforming the baseline (Claude Code with direct repository access) at 80.3 ± 2.3% ($p < 0.0001$, bootstrap test; 95% CI of difference: [5.8%, 15.8%]). Paper2Agent answered benchmark questions at an average cost of $0.20 per query in 1.6 minutes, compared to $0.38 per query in 4.3 minutes for the baseline, corresponding to a 1.9x cost reduction and 2.7x speedup. For the remaining 26 computational papers that could not be fully agentified, dominant failure modes included missing executable code (35%), missing data or model artifacts (31%), environment or dependency failures (23%), and non-generalizable scripts (12%). These failures strongly correlate with documentation completeness, environment specification quality, and the presence of executable tutorials.

"Even when executable tools cannot be constructed, Paper2Agent still provides value through its structured resource layer. Across 26 non-computational papers (13 bioRxiv and 13 Nature papers), we curated 100 synthesis-based benchmark questions spanning main text and supplementary materials. Paper2Agent achieved 89.0 ± 3.1% accuracy, significantly outperforming a strong Claude browser-use baseline (82.0 ± 3.8%, $p = 0.03$), while being 34× cheaper and 15× faster. To understand whether the agent will reject out-of-scope queries, we constructed an adversarial out-of-scope benchmark by randomly permuting paper–question pairs across the same 26 papers. Under instructions to answer only using local paper files and respond "I don't know" otherwise, Paper2Agent achieved a 100% correct rejection rate, reliably declining to answer mismatched paper–question pairs."

3. Accuracy is reported as a single point estimate (on AlphaGenome queries) based on manual grading. There are no confidence intervals, no assessment of inter-rater agreement, and no robustness analysis with respect to prompt phrasing or repeated runs. The manuscript acknowledges these limitations, but the conclusions lean on strong reliability language. At minimum, the authors should report confidence intervals for success rates, run-to-run variance, and evaluate sensitivity to paraphrased queries.

Response: We thank the reviewer for this suggestion. We have added the analysis that ran each method five times [5 runs × (15 + 15) questions = 150 evaluations per method] and reported mean accuracy ± SE across runs.

Method	Tutorial tasks (mean ± SE)	Novel tasks (mean ± SE)
AlphaGenome Agent	98.7% ± 1.3%	100.0% ± 0.0%

Claude + repo	82.7% ± 3.4%	78.7% ± 4.4%
Biomni	37.3% ± 4.0%	56.0% ± 3.4%

To assess whether AlphaGenome agent significantly outperforms the baselines, we conducted one-sided paired t-tests on per-run accuracy. AlphaGenome agent significantly outperformed both Claude + repo (tutorial: $p = 8.1 \times 10^{-3}$; novel: $p = 4.2 \times 10^{-3}$) and Biomni (tutorial: $p = 4.7 \times 10^{-5}$; novel: $p = 1.0 \times 10^{-4}$). Besides, we further evaluated the AlphaGenome agent using 30 paraphrased queries, and it gave similar accuracy (tutorial: 98.7% ± 1.3%; novel: 97.3% ± 1.6%).

This is consistent with the original conclusion of the paper showing that the AlphaGenome agent generated by Paper2Agent outperforms alternative approaches. For other newly added benchmark results, we have also added the mean ± SE for the reported accuracy.

We have added these results to the Results section of the revised manuscript.

Revised Method section:

"Across five independent runs, the AlphaGenome agent achieved 98.7% ± 1.3% accuracy on these queries, significantly higher than Claude + Repo (82.7% ± 3.4%, $p = 8.1 \times 10^{-3}$) and Biomni (37.3% ± 4.0%, $p = 4.7 \times 10^{-5}$)."

"Across five independent runs, the AlphaGenome agent achieved 100.0% ± 0.0% accuracy, as verified by manual execution of the original AlphaGenome code and evaluated by predefined rubrics. This is significantly higher than Claude + Repo (78.7% ± 4.4%, $p = 4.2 \times 10^{-3}$) and Biomni (56.0% ± 3.4%, $p = 1.0 \times 10^{-4}$)."

"To assess robustness to prompt variation, we further evaluated the AlphaGenome agent on 30 paraphrased versions of the tutorial and novel queries. The agent achieved similar accuracy (tutorial: 98.7% ± 1.3%; novel: 97.3% ± 1.6%), demonstrating that performance is not sensitive to specific query phrasings."

4. Paper2Agent depends on the quality and completeness of the underlying codebase, and that not all papers can be faithfully converted into agents. If this is the case and the agent is not useful for papers without complete codebases and documented tutorials, then the added advantage may be minimal.

Response: We thank the reviewer for the suggestion. We address it from two angles: robustness to imperfect codebases, and the value Paper2Agent provides even when full agentification is not possible.

First, to evaluate robustness to codebase defects, because repositories with known, well-documented issues are difficult to identify systematically, we manually injected common

execution-level errors into three representative repositories spanning different programming paradigms: AlphaGenome (Python-based), POP-TOOLS (command-line–based), and mlearner (R-based). For each repository, we considered four adversarial configurations that simulate realistic implementation failures: (1) removal of required dependencies from environment specifications (e.g., requirements.txt), (2) modification of input file paths to nonexistent locations, (3) insertion of cell-level execution errors into notebooks or tutorials, and (4) replacement of API calls with deprecated equivalents when applicable. This resulted in a total of 12 adversarial case studies. In all cases, Paper2Agent's iterative execute-diagnose-repair loop detected runtime failures and applied bounded, local repairs before re-running. Across all 12 configurations, end-to-end execution succeeded with a 100% success rate, demonstrating that Paper2Agent can tolerate and recover from common codebase defects rather than requiring pristine repositories.

Second, we acknowledge that not all papers can be fully agentified. In our large-scale evaluation of 100 randomly selected computational papers from bioRxiv, 26 papers (26%) could not be converted into executable agents. Failure modes included missing executable code (35% of failures), missing data or model artifacts (31%), unresolvable environment dependencies (23%), and non-generalizable scripts (12%). Critically, these failures reflect fundamental limitations of the underlying repositories — incomplete code, unreleased data, or hardcoded reproduction scripts — rather than limitations of the Paper2Agent pipeline itself. Many of these repositories would also be unusable by a human researcher without significant additional effort from the original authors.

However, even for these papers, Paper2Agent still provides meaningful value through its structured resource layer, which exposes the manuscript text, supplementary materials, and metadata for retrieval-augmented question answering. On 100 synthesis-based benchmark questions across 26 non-computational papers, Paper2Agent achieved 89.0 ± 3.1% accuracy, significantly outperforming a Claude browser-use baseline (82.0 ± 3.8%, $p = 0.03$), while being 34× cheaper and 15× faster. This demonstrates that Paper2Agent's utility extends well beyond executable tool construction: for all papers, regardless of code quality, it provides a structured, queryable interface to the paper's content and findings.

Together, these results show that Paper2Agent is robust to imperfect codebases when executable code exists, and remains useful through its resource layer when it does not. The failure modes we identified can inform community standards for code sharing practices. We have revised the Results and Discussion sections of the manuscript to reflect these findings.

We have added those new results in the Results and Discussion section of the revised paper:

Revised Results section:

“We also conducted adversarial stress tests of Paper2Agent to assess robustness to repository defects and out-of-scope query scenarios. To test robustness to upstream code failures, we constructed adversarial variants of three representative repositories spanning different programming environments: AlphaGenome (Python-based), POP-TOOLS (Python

command-line-based), and mlearner (R command-line-based). For each repository, we injected four categories of execution-level errors: (1) removal of required dependencies from environment specifications, (2) modification of input file paths to nonexistent locations, (3) insertion of typographical errors into code, and (4) replacement of API calls with deprecated equivalents, yielding 12 adversarial configurations in total. In each case, the agent's iterative execution loop detected runtime failures from error tracebacks, applied targeted local fixes, and re-executed the workflow. Across all 12 configurations with three independent runs, Paper2Agent completed end-to-end successfully (36/36), producing functional MCP servers that successfully reproduced the tutorial results."

"Next, we conducted a large-scale evaluation across a heterogeneous corpus of research papers. We randomly selected 100 computational research papers from bioRxiv's bioinformatics category, 26 general research papers from bioRxiv and Nature that focus on data and discovery rather than executable methods, and 5 computational papers from non-biology domains including AI and statistics. All papers were processed end-to-end by Paper2Agent without manual cleanup, code modification, or intervention."

Paper2Agent produces two complementary components for each paper: (i) a structured resource layer that exposes the manuscript text, supplementary information, and metadata for retrieval and question answering, and (ii) an executable tool layer that exposes computational methods as callable MCP tools. While the resource layer is always constructible, successful tool-layer construction depends on the quality and agentifiability of the associated code repository.

"Among the 100 computational papers, 74 of 100 (74%) were successfully agentified with executable tools. Paper2Agent proposed 599 tools in total, of which 593 passed automated validation. The average cost for processing one successfully agentified paper was $14.99 with an average runtime of 2.4 hours; papers that failed typically failed early in the pipeline (average cost $9.03, average time 1.7 hours). The generated MCP server implementations were further inspected for potential shortcut learning or tutorial-specific memorization. We did not find systematic evidence of hard-coded tutorial constants, fixed dataset paths, cached outputs, or dataset-specific heuristics.

We evaluated these 74 paper agents on 300 benchmark questions derived from GitHub tutorials. Paper2Agent achieved 91.2 ± 1.6% accuracy (mean ± bootstrap SE), significantly outperforming the baseline (Claude Code with direct repository access) at 80.3 ± 2.3% ($p < 0.0001$, bootstrap test; 95% CI of difference: [5.8%, 15.8%]). Paper2Agent answered benchmark questions at an average cost of $0.20 per query in 1.6 minutes, compared to $0.38 per query in 4.3 minutes for the baseline, corresponding to a 1.9x cost reduction and 2.7x speedup. For the remaining 26 computational papers that could not be fully agentified, dominant failure modes included missing executable code (35%), missing data or model artifacts (31%), environment or dependency failures (23%), and non-generalizable scripts (12%). These failures strongly correlate with documentation completeness, environment specification quality, and the presence of executable tutorials.

Even when executable tools cannot be constructed, Paper2Agent still provides value through its structured resource layer. Across 26 non-computational papers (13 bioRxiv and 13 Nature papers), we curated 100 synthesis-based benchmark questions spanning main text and supplementary materials. Paper2Agent achieved 89.0 ± 3.1% accuracy, significantly outperforming a strong Claude browser-use baseline (82.0 ± 3.8%, p = 0.03), while being 34× cheaper ($0.27 vs $9.04 per query) and 15× faster (63s vs 919s per query).”

Revised Discussion:

“Not every paper can be seamlessly turned into a robust agent. As we demonstrate in the adversarial benchmarks, Paper2Agent’s iterative execute-diagnose-repair loop can detect and correct many inherited bugs that manifest as execution failures. Nevertheless, in our large-scale evaluation, a substantial fraction of repositories still could not be successfully agentified, typically due to incomplete codebases, missing documentation, or unresolvable environment configurations. Moreover, bugs that do not trigger runtime exceptions but instead produce silently incorrect outputs, such as a flawed formula that returns plausible but wrong results, remain fundamentally difficult to detect without external ground-truth references, a limitation shared by any system that relies on execution-based validation. These challenges suggest that the ease with which a paper can be transformed into an agent may itself serve as a practical measure of reproducibility and rigor. Just as the scientific community has come to expect clear data and code availability, we envision a natural extension: expecting contributions to be structured in ways that facilitate their translation into agents. Well-documented, modular, and transparent papers will naturally lend themselves to this new standard.”

Minor points

1. The definition of “novel queries” should be clarified operationally

Response: We thank the reviewer for this suggestion. We have clarified the operational definition of novel queries in the Results section. Specifically, novel queries are defined as evaluation queries that are not present in the GitHub tutorial notebooks, and that require executing the paper’s tools on new inputs or parameter combinations beyond those demonstrated in the tutorials.

Revised Results section:

“To assess generalizability and guard against potential overfitting to the original examples, we manually curated a set of novel queries that were not present in the GitHub tutorials, and that require applying the method to new inputs or parameter settings beyond those demonstrated in the original tutorials.”

2. A rubric for manual grading would improve transparency

Response: We thank the reviewer for this suggestion. We have added the rubric in the Supplementary Notes section.

3. The Scanpy and TISSUE examples would benefit from even small quantitative evaluations

Response: We thank the reviewer for this suggestion. In the revised manuscript, we have added quantitative evaluations for both the Scanpy and TISSUE case studies by comparing key numerical outputs from the agents (e.g., the number of cells and genes retained after quality control, top differentially expressed genes, and upper bounds of predictive intervals in tissue prediction) with those obtained from manual execution of the original pipelines. We find that the results are highly consistent between human and agent. We have revised the Results section in the manuscript to reflect this:

Revised Results section:

"As shown in Figure 4C, the agent produces outputs that match those produced by human researchers when processing the same data, retaining equivalent cell and gene counts after quality control and recovering equivalent top differentially expressed marker genes per cluster with matched parameters."

"The output matches results obtained by human experts running the pipeline manually, producing equivalent prediction interval bounds and concordant figure outputs across 5 independent runs"

4. Clarify what is a passing test (Fig. 1B) and what is the policy when failures persist

Response: We thank the reviewer for this suggestion. We have clarified the testing criteria and failure policy in the revised Results section. A test passes when numerical results fall within tolerance thresholds and generated figures match reference visualizations from the original tutorials. When failures persist after multiple repair attempts (6 in our current setting), the tool is excluded from the final MCP server, ensuring that only validated tools are exposed to users.

Revised Results section:

"A test passes when expected files are generated, numerical results fall within tolerance thresholds, and figures match references; tools that repeatedly fail validation are excluded from the final MCP server."

Revised Extended Method section:

"Test-verifier-improver: a specialized agent that creates, runs, and refines tests for tutorial implementations. It uses only the tutorial's own examples to ensure complete coverage and faithful reproduction of numerical and visualization results. A test passes when expected files are generated, numerical results match tutorial outputs exactly (with 3% tolerance for

floating-point values), and generated figures match reference visualizations (verified via perceptual hashing with Hamming distance < 20). The agent runs in a loop of generating tests, executing them, diagnosing failures, and applying fixes, with a maximum of 6 attempts per function. If functions repeatedly fail, their MCP decorators are removed, a failure comment is added, and they will not be included in the MCP server. All results and logs are recorded for transparency."

5. What are hardware specs for runtime test (Fig. 2C)

Response: We thank the reviewer for raising this point. All runtime experiments in Fig. 2C were conducted on a MacBook Air equipped with an Apple M2 chip (8 core CPU, 8 core GPU, 8 GB unified memory), using model APIs as needed. We have now added the full hardware specifications to the Extended Methods section.

Revised Extended Method section:

"For all agent evaluations, we used claude-sonnet-4-20250514 as the underlying LLM. All evaluations were run locally on a MacBook Air (M2 chip) (8-core CPU, 8-core GPU, 8 GB unified memory), using model APIs as needed."

6. Fig 2D: Include summary statistic: e.g., number of loci/tasks where the full report generation completed without manual intervention, and common failure modes.

Response: We thank the reviewer for this suggestion. Fig. 2D is intended as an illustrative case study focusing on a single LDL GWAS locus. To evaluate robustness, we reran the AlphaGenome agent 10 times on this locus and observed no failures. The agent successfully generated complete multi-modal interpretation reports without manual intervention in all runs. In a separate case study, we applied the agent to 37 independent ADHD GWAS loci, all of which also completed successfully without errors.

7. Fig 3C: how do you quantify matching? Report quantitative similarity and benchmarks. Are these results reproducible?

Response: We thank the reviewer for this suggestion. We have added quantitative similarity metrics to evaluate the agreement between figures generated by the human and the agent. Specifically, we compute perceptual similarity between the rendered figures using perceptual hash distance, and additionally report the similarity between the underlying numerical outputs, including both the upper and lower bounds of the predictive intervals.

We observed that the quantitative similarity between the human- and agent-generated figures is consistently high across runs, with low perceptual hash distances and strong numerical agreement in the reported predictive intervals. We have revised our manuscript to reflect this.

Revised Results section:

“The output matches results obtained by human experts running the pipeline manually, producing equivalent prediction interval bounds and concordant figure outputs across 5 independent runs”

Referee #3 (Remarks on code availability):

The repository provides a README with a clear high-level description, example commands, and a shell-based entry point that documents how to run the pipeline on a target GitHub repository, including options for tutorial filtering and benchmarking, which is helpful for technically experienced users.

Referee #4 (Remarks to the Author):

I co-reviewed this manuscript with one of the reviewers who provided the listed reports.

Referee #5 (Remarks to the Author):

I co-reviewed this manuscript with one of the reviewers who provided the listed reports.

Reviewer #1 (Remarks to Authors):

The authors have performed a substantial revision, adding a 100-paper evaluation, ablation studies, cross-platform testing, and a new discovery case study. I acknowledge the effort and recognize that many specific requests from the first round have been addressed with new data. However, the field of AI agents has moved rapidly since the original submission, and I have significant concerns about the durability and significance of the contribution.

Improvements:

The large-scale evaluation (74/100 papers successfully agentified, 91.2% accuracy on 300 benchmark questions) with detailed failure mode characterization provides the systematic scope assessment that was previously missing. The ablation studies convincingly justify specific architectural choices — the monolithic agent's complete failure (0/3 runs), the accuracy drop without test-verification (from ~99% to ~69%), and the MCP vs. skill file gap (99.3% vs 73.3%) demonstrate that each design choice contributes meaningfully. The OpenCode backend evaluation (98.7% tutorial, 97.3% novel accuracy) establishes that Paper2Agent is not inherently Claude-specific, addressing a major generalizability concern from the first round. The GPR137/psoriasis case study adds a validation component using existing scCRISPRi and Perturb-seq data, though as noted below this remains a computational analysis rather than independent experimental validation. Statistical rigor is now appropriate throughout, with bootstrap SEs, confidence intervals, and paired significance tests across substantially expanded sample sizes.

Response:
We sincerely thank the reviewer for the thoughtful feedback and for recognizing that our revision has addressed many of the requests from the first round. Your suggestions have truly helped us to further strengthen the manuscript.

Remaining concerns:

1. General-purpose coding agents are increasingly capable of doing on-the-fly what Paper2Agent pre-computes. The paper's own data illustrates this: Claude Code with direct repository access already achieves 80.3% accuracy on the large-scale benchmark, compared to Paper2Agent's 91.2%. This is an 11 percentage point gap, real but modest. The baseline was measured using claude-sonnet-4-20250514, a model now several generations behind current frontier capabilities. The gap has almost certainly narrowed since the experiments were run and will likely continue to shrink as models improve.

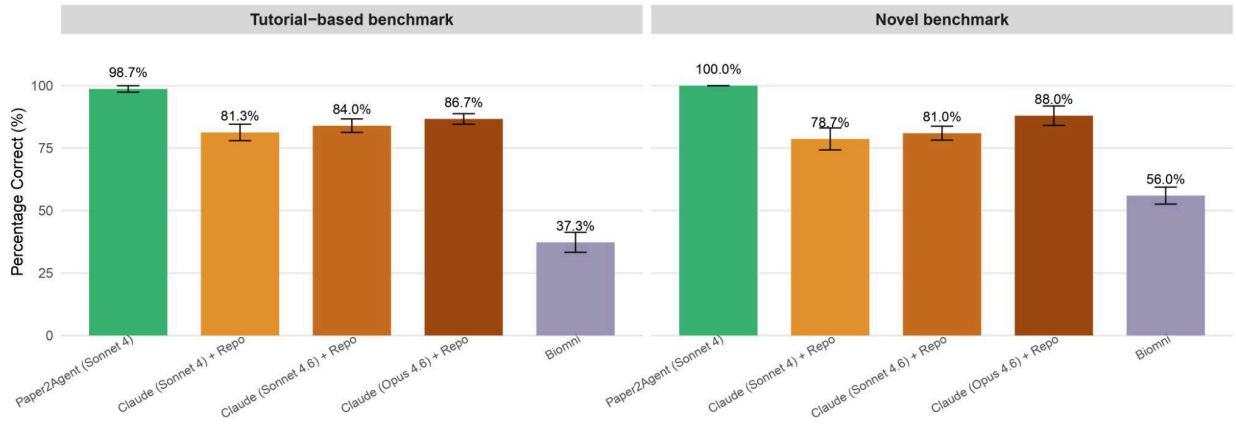
The economics reinforce this concern. At ~$15 and 2.4 hours per paper on average, with cost break-even at ~245 queries, most niche computational tools will never reach the threshold where pre-building an MCP server pays off. The speed advantage (1.6 vs 4.3 min/query) is real but increasingly marginal as inference costs drop. The authors frame agentification as a "one-time community investment," but this requires adoption infrastructure (journals or

communities maintaining agents) that does not yet exist and whose feasibility is undemonstrated.

More fundamentally, a pre-computed MCP server is a frozen artifact that decays as upstream codebases evolve (API changes, dependency updates, data format shifts). The authors acknowledge this and propose maintenance agents (Claude Code Web/Jules) for periodic re-validation — but this concedes that the MCP server requires ongoing energy to prevent decay. A general-purpose agent interacting with the live repository requires no maintenance and always operates on the current codebase. The maintenance burden may ultimately undermine the efficiency gains that justify the approach.

Response:

Thank you for this comment. We address this concern directly by adding updated Claude + Repo baselines using Claude Sonnet 4.6 and Claude Opus 4.6. On the AlphaGenome benchmark, Claude Sonnet 4.6 + Repo achieved 84.0% ± 2.7% accuracy on tutorial-based queries and 81.0% ± 2.8% on novel queries, and Claude Opus 4.6 + Repo achieved 86.7% ± 2.1% (tutorial) and 88.0% ± 3.9% (novel), compared with Paper2Agent at 98.7% ± 1.3% (tutorial) and 100.0% ± 0% (novel). The accuracy gap remains substantial, about 12 percentage points on both benchmarks, even when the baseline is upgraded to frontier coding models that were released after our previous revision. Moreover, this stronger baseline comes at over double the per-query cost of Paper2Agent ($0.36 per query for Paper2Agent vs $0.74 for Opus 4.6).



Response Fig. 1 (new Supp. Figure 2). Paper2Agent maintains an accuracy advantage over frontier coding agents with direct repository access in tutorial and novel benchmarks.

In addition, Paper2Agent's pre-build test–verifier–improver layer provides an error-checking step that is not available when an LLM is given a repository to use directly. Because Paper2Agent uses subagents to test the paper's codebase to reproduce results explicitly, it's able to catch and repair mistakes in the repository. Directly asking a frontier LLM to use the repository without our reproduction tests can miss such mistakes.

As an example, we asked Claude Code with Opus 4.6 to run POP-GWAS, a published machine-learning-assisted GWAS method, on its bundled `Head_BMD` test data[1]. The package contains latent NumPy and Python 3.12 incompatibilities that cause the pipeline to silently produce an all-NA output while exiting cleanly. Opus 4.6 diagnosed the bugs, but in the process also took a shortcut that bypassed one of POP-GWAS's statistical correction steps, substituting hardcoded values for quantities the algorithm was meant to estimate from the data. The output looked successful, but the reported top hit became rs62228067 at chr22:46.4 Mb with P = 1.8e-64, which is incorrect. The correct analysis identified rs41158 at chr22:30.4 Mb with P = 2.5e-7, a different locus and more than fifty orders of magnitude different in statistical significance. By contrast, during our previous revision, we used Paper2Agent to build a POP-TOOLS MCP server. Paper2Agent's test–verifier–improver flagged the same silent failure on the bundled CLI test, applied the minimal compatibility patches needed to pass the test, and verified that the bundled test then reproduced the published result. We subsequently merged these three patches into upstream POP-TOOLS (commits `5cbcb63`, `4c5259e`, `f1db403`) to fix this issue, providing a real-world example of Paper2Agent enhancing reproducibility of a method for broader community benefit.

We have further revised the maintenance discussion to emphasize the conceptual shift we are advocating. Paper agents do require maintenance, but this is also true of code, containers, notebooks, and hosted reproducibility platforms. Our view is that agent-native artifacts should become part of what authors publish and maintain, just as many researchers now maintain GitHub repositories, documentation, containers, and example notebooks after publication. We will present this as a necessary part of agent availability, not as a reason to abandon agentification.

We have revised our manuscript to reflect those:

Revised Results section:

"These gains were robust to prompt paraphrasing and persisted when the Claude + Repo baseline was upgraded to newer chat models (Supplementary Figures 2 and 3, Supplementary Note)."

Revised Supplementary Note section:

"This advantage does not vanish when the chat model is updated. Upgrading the Claude + Repo baseline from Sonnet 4 to Sonnet 4.6 (tutorial: 84.0% ± 2.7%; novel: 81.0% ± 2.8%) and to Opus 4.6 (tutorial: 86.7% ± 2.1%; novel: 88.0% ± 3.9%) narrowed the gap only modestly relative to Paper2Agent's 98.7% ± 1.3% / 100.0% ± 0.0% (Supplementary Figure S2). On open-ended benchmarks, Paper2Agent retained a 12-14 percentage-point gap over Claude + Repo at every chat-model generation tested, with even Paper2Agent (Sonnet 4) at 82.7% ± 2.4% exceeding Opus 4.6 + Repo at 81.3% ± 2.0% (Supplementary Figure S3), while Paper2Agent (Opus 4.6) achieves 93.3% ± 2.1%."

Revised Discussion section:

"More broadly, Paper2Agent advocates a conceptual shift in how scientific knowledge is represented and reused: papers become agent-native research objects rather than static documents."

Revised Introduction section:

"This extends beyond retrieval-augmented generation over paper text. Paper2Agent agentifies the full publication ecosystem, including manuscripts, supplementary materials, code, datasets, executable examples, and analysis workflows. In this framework, a paper becomes an executable research artifact that can answer questions, reproduce analyses, apply methods to new data, and interoperate with other paper agents."

2. Related to the above, the new large-scale evaluation data suggests the contribution may be closer to optimized retrieval-augmented generation (RAG) than a paradigm shift. The structured resource layer (for non-computational papers) achieves 89.0% vs 82.0% for a Claude browser-use baseline, a 7 percentage point gap. The executable tool layer adds more value (91.2% vs 80.3%), but the multi-agent conversion pipeline is fundamentally a wrapper around the underlying LLM's coding ability. It does not introduce new algorithms, new scientific reasoning, or new domain knowledge; it reorganizes existing code into a format that is easier for an LLM to invoke. As coding agents improve at reading documentation and executing code on first attempt, this reorganization becomes less necessary. The presented data does not suggest that Paper2Agent adds any major independent capability beyond what the base model provides, and the value of the wrapper will erode as the base models catch up.

Response: Thank you for this comment. While Paper2Agent delivers measurable engineering gains in accuracy and latency, it also has a fundamental conceptual contribution: it redefines how scientific knowledge is represented and used in the era of AI agents.

For centuries, the scientific paper has been a static medium, requiring human readers to interpret methods, reconstruct workflows, and manually operationalize results. Paper2Agent shifts to an agent-native representation, where a paper is instantiated as a virtual corresponding author that helps disseminate and apply its findings in new settings. In this paradigm, papers become active knowledge rather than passive documents: they can answer questions, execute analyses, reproduce results, and interact with other paper agents to enable autonomous, cross-study discoveries.

Paper2Agent is also more than an improved RAG system over paper text. The goal is to agentify the full set of artifacts associated with a paper, including the manuscript, supplementary materials, datasets, code repository, executable examples, validation tests, and scientific workflows. In typical paper-QA or RAG settings, the agent often retrieves from a narrow textual representation of a paper, sometimes only the abstract or main text.

This distinction is reflected in empirical results. For non-computational papers using the structured resource layer, Paper2Agent improved accuracy over the Claude browser-use baseline (89.0% vs. 82.0%), which is a substantial gain for questions requiring synthesis across main text and supplementary materials. The cost and latency improvements were even larger: Paper2Agent answered questions at $0.27 per query in 63 seconds, compared with $9.04 per query and 919 seconds for the Claude browser-use baseline, corresponding to an approximately 34x cost reduction and 15x speedup.

We have further clarified why Paper2Agent's value should not simply erode as base models improve. A general-purpose coding agent may solve a repository task once, but that one-off interaction does not create a shared publication artifact with stable interfaces, validation tests, provenance, known limitations, and maintenance hooks, as we showed in response to the first comment above. Instead, stronger agents make the Paper2Agent vision more feasible: they can improve the construction, maintenance, and composition of paper agents at scale.

3. The authors clarify that Paper2Agent does not access the manuscript during evaluation, only the pre-built MCP tools and metadata. The "Claude + Repo" baseline receives both the full code repository and the paper. This means the baseline has equivalent raw information access, yet Paper2Agent still outperforms it by 11 points. The gain therefore stems from the pre-computed tool extraction and validation : the codebase has been distilled into structured, pre-tested API calls. While this is a legitimate engineering advantage, it is the kind of advantage that general-purpose agents will increasingly replicate on the fly as they become better at reading documentation and executing code correctly on first attempt. This reviewer's own experience is that current frontier models (e.g., Claude Opus 4.6) can already read a paper's repository, understand its API, and execute correct tool calls without pre-computed scaffolding, suggesting the baseline used in this study substantially underestimates current general-purpose agent capability.

Response: Thank you for this comment. Our results show a substantial gap between giving an agent a human-oriented paper/repository and giving it an agentified paper interface. The observed gain measures the value of agentification itself: converting the same underlying paper and repository into structured tool interfaces, workflow prompts, provenance links, and pre-validated executable functions.

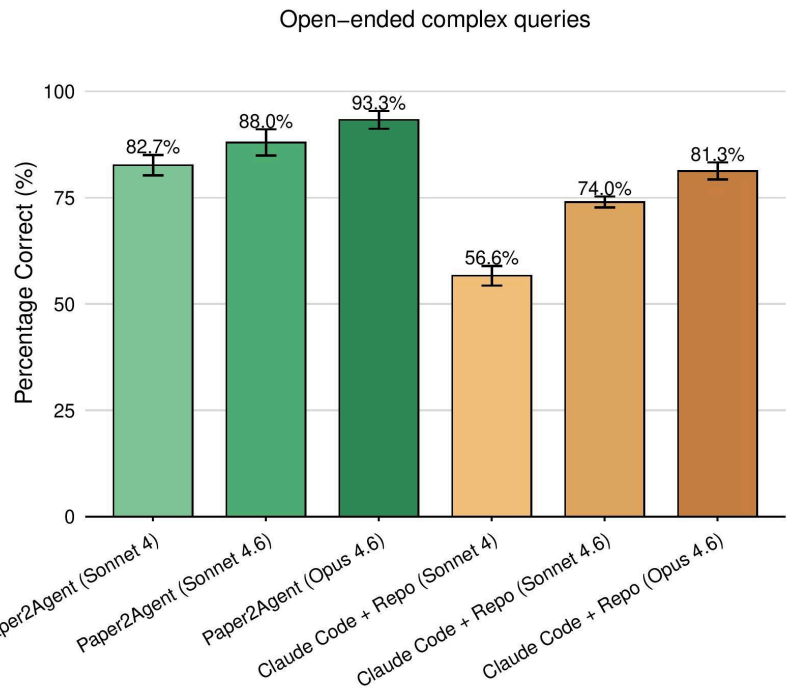
More broadly, we believe scientific repositories and paper artifacts should increasingly be designed in agent-native formats. Current repositories are usually optimized for human developers reading documentation, notebooks, and scripts. Our goal with Paper2Agent is to show the benefits of this agent-native format: papers should be released with paper agents, including MCP servers where appropriate, in the same way that papers are increasingly released with code and data. These paper agents can expose interfaces designed for agents, such as structured tool schemas, validation tests, usage boundaries, and provenance links. This makes the work easier for both humans and agents to reuse, audit, and maintain.

4. The authors inspected MCP implementations for hardcoded artifacts and added an automated reviewer agent. However, the fundamental design (tools are built and validated against tutorials, then evaluated on queries derived from those same tutorials) means the training and evaluation distributions remain coupled. The 100% accuracy on "novel" AlphaGenome queries remains statistically suspicious; in real-world scientific use, "novel" means unforeseen edge cases, and perfect accuracy on 15 queries suggests the benchmark is still too close to the training distribution. The open-ended queries (84% accuracy on 15 queries for a single case study) provide the strongest evidence against pure pattern matching but are too limited to establish robust generalization.

Response: Thank you for this comment. We want to clarify that these benchmarks reflect realistic paper-use cases, not artificial memorization tests. In practice, many users first approach a computational paper by asking the agent to reproduce, adapt, or extend the analyses demonstrated in the paper's tutorials and examples. These are precisely the workflows that determine whether a method is usable after publication. Importantly, our results show that even these realistic usage tasks are not solved reliably by general-purpose agents with repository access, whereas Paper2Agent improves accuracy, cost, and latency.

The broader evaluations already show that the assessment is not limited to exact tutorial reproduction. These include 15 open-ended AlphaGenome queries requiring multi-step reasoning, tool composition, and biological synthesis; 300 benchmark questions across 74 computational biology papers; 100 synthesis-based questions across 26 non-computational papers. We also evaluated non-biology computational papers with tasks beyond simple reproduction, including SAELens sparse autoencoder training with iterative hyperparameter adaptation and TabPFN model training/comparison using cross-validation.

To further address the reviewer’s concern, we added 15 additional open-ended queries for AlphaGenome that are not derived from the tutorials and instead require tool composition and analytical reasoning. These new queries focus on disease-associated variant interpretation in previously unseen contexts. They require the agent to score variants, identify potential regulatory mechanisms, and synthesize evidence across multiple modalities. Importantly, these tasks reflect the real-world use cases for AlphaGenome encountered by practicing scientists. This leads to a total of 30 open-ended questions. On the combined 30 open-ended question benchmark, Opus 4.6 + Paper2Agent MCP achieves 93.3% ± 2.1% versus 81.3% ± 2.0% for repository-only access (also with Opus 4.6), a 12.0-point gap that is more than triple the noise level. The same pattern holds for Sonnet 4.6 + Paper2Agent MCP, which achieves 88.0% ± 3.1% versus 74.0% ± 1.3% for Sonnet 4.6 + Repository, a 14.0-point gap. The Paper2Agent advantage is therefore not an artifact of an oversaturated benchmark; the gap is largest on precisely the questions that move farthest from tutorial demonstrations.



Response Fig. 2 (new Supp. Figure 3). Paper2Agent maintains an accuracy advantage over frontier coding agents with direct repository access on open-ended complex queries.

We have revised our manuscript to reflect those:

"We further evaluated the AlphaGenome agent on 30 open-ended queries reflecting realistic researcher usage."

"Paper2Agent achieves superior performance (82.7% ± 2.4%) compared with Claude + Repo (56.7% ± 2.3%, p = 5.6e-5) and Biomni (72.2% ± 2.2%, p = 4.3e-4) across five independent runs, while also being more efficient (5.3 min, $0.38 per query) than Claude + Repo (8.3 min, $0.52 per query) and Biomni (11.2 min, no cost data available)."

5. In response to my original request for evidence that the Scanpy agent adapts pipeline parameters based on dataset characteristics, the revised manuscript claims that "the agent adaptively adjusts parameters based on data characteristics: selecting organism-appropriate mitochondrial gene prefixes, reducing HVG counts for targeted gene panels, and applying tissue-specific marker genes." This is asserted in a single sentence without systematic evidence. A supplementary table showing specific parameter choices across diverse datasets would be needed; as written, it could equally reflect the agent following a rigid template that happens to work on the tested datasets.

Response: Thanks for the reviewer's suggestion. We have expanded the analysis to seven diverse single-cell datasets spanning two species and three tissue families (immune, neural, cardiac), including a disease-state dataset (brain tumor), and added a new Supplementary Table documenting the Scanpy agent's per-dataset parameter choices. The seven datasets are

three human PBMC, one human brain tumor, two mouse brains at different scales, and one mouse heart. Each dataset was run end-to-end through the Scanpy MCP agent, starting from the same generic prompt with no dataset-specific hints. The Supplementary Table reports, for each dataset, the agent's observed evidence, selected parameters at each pipeline step, and recorded rationale, captured directly from the MCP tool-call logs.

The Supplementary Table documents distinct adaptive behaviors:

(i) Organism-specific mitochondrial prefix. The agent inferred organism from gene-symbol case in `var_names` and applied `MT-` to all four human datasets and `mt-` to all three mouse datasets, without user specification.

(ii) Tissue-specific marker-gene panels. The agent applied canonical PBMC immune markers (`CD3D`, `MS4A1`, `CD14`, `NKG7`) for the human-PBMC datasets; mouse neural lineage markers (`Slc17a7`, `Gad1/Gad2`, `Gfap`, `Mbp`) for the mouse-brain datasets; and mouse cardiac markers (`Myh6`, `Tnnt2`, `Col1a1`, `Pecam1`) for the mouse-heart dataset. No mouse marker appeared in a human-PBMC run, and no PBMC marker appeared in a mouse run; the species/tissue boundary of the marker panel was preserved across all datasets.

(iii) Disease-state recognition. For the brain tumor dataset, the agent supplemented canonical brain markers with glioma drivers (`EGFR`, `PDGFRA`, `CDK4`), tumor-associated macrophage markers (`AIF1`, `CX3CR1`, `P2RY12`), and infiltrating T-cell markers (`CD3D`, `CD3E`), reflecting the malignant context rather than treating it as healthy brain tissue.

Behavior (iii) in particular cannot be produced by a fixed template: a template cannot semantically recognize a disease context and pull glioma-specific drivers into the marker panel.

We have attached the new supplementary table:

Supplementary Table 2. Per-dataset parameter choices selected by the Scanpy agent across seven single-cell datasets. The agent received the same generic prompt for every dataset; the inferred organism, mitochondrial gene prefix, and marker gene panel were determined directly from data without user specification. Columns: `dataset` (dataset name); `organism_inferred` (organism inferred by the agent from gene-symbol case); `tissue` (tissue or cell-type context); `gene_panel_context` (post-filtering gene/cell counts observed by the agent); `agent_observed_evidence` (evidence the agent cited for its organism call); `step1_mt_prefix_selected` (mitochondrial prefix the agent applied at the QC step); `step7_marker_gene_panel` (marker genes the agent selected for cell-type annotation); `agent_rationale` (the agent's recorded justification). All entries are captured verbatim from MCP tool-call logs.

dataset	organism_inferred	tissue	gene_panel_context	agent_observed_evidence	step1_mt_prefix_selected	step7_marker_gene_panel	agent_rationale

pbmc_1k_v3	Human	PBMC (10x Genomics)	15,247 genes post-filtering	Human gene name pattern	MT-	CD3D, CD3E for T cells; MS4A1 for B cells; CD14 for monocytes; NKG7 for NK cells	The agent inferred the organism from gene name patterns and correctly identified and applied the MT- prefix to label mitochondrial genes; the agent applied canonical immune cell markers
pbmc_1k_v2	Human	PBMC (10x Genomics)	33,538 genes	Human gene name pattern	MT-	CD3D, CD3E for T cells; MS4A1 for B cells; CD14 for monocytes; NKG7 for NK cells	The agent inferred the organism from gene name patterns and correctly identified and applied the MT- prefix to label mitochondrial genes; the agent applied canonical immune cell markers
pbmc_1k_protein_v3	Human	PBMC (10x Genomics CITE-seq)	33,538 genes (RNA modality)	Human gene name pattern	MT-	CD3D, CD3E for T cells; MS4A1 for B cells; CD14 for monocytes; NKG7 for NK cells	The agent inferred the organism from gene name patterns and correctly identified and applied the MT- prefix to label mitochondrial genes; the agent applied canonical immune cell markers
Brain_Tumor_3p_LT	Human	Brain tumor (10x Genomics)	36,601 genes (182 cells)	Human gene name pattern	MT-	EGFR, PDGFRA, CDK4 for tumor/glioma cells; GFAP, AQP4 for astrocytes; MBP, MOG for oligodendrocytes; AIF1, CX3CR1, P2RY12 for microglia/tumor-associated macrophages; CD3D, CD3E for infiltrating T cells	The agent inferred the organism from gene name patterns and correctly identified and applied the MT- prefix to label mitochondrial genes; the agent supplemented brain markers with glioma drivers and tumor-associated macrophage markers, reflecting the malignant context

neuron_1k_v3	Mouse	Brain (E18 cortex; 10x Genomics)	31,053 genes (1,301 cells)	Mouse gene nomenclature conventions	mt-	Slc17a7, Slc17a6 for excitatory neurons; Gad1, Gad2 for inhibitory neurons; Gfap, Aqp4 for astrocytes; Mbp, Mog for oligodendrocytes ; Pdgfra for oligodendrocyte precursor cells	The agent automatically switched to the mt- prefix, recognizing the organism from gene nomenclature conventions without explicit user specification; the agent automatically switched to brain-specific neural lineage markers
neuron_10k_v3	Mouse	Brain (cortex; 10x Genomics)	31,053 genes (11,843 cells)	Mouse gene nomenclature conventions	mt-	Slc17a7, Slc17a6 for excitatory neurons; Gad1, Gad2 for inhibitory neurons; Gfap for astrocytes; Mbp, Mog, Plp1 for oligodendrocytes ; Pdgfra for OPCs	The agent automatically switched to the mt- prefix, recognizing the organism from gene nomenclature conventions without explicit user specification; the agent automatically switched to brain-specific neural lineage markers
heart_1k_v3	Mouse	Heart (10x Genomics)	31,053 genes (1,011 cells)	Mouse gene nomenclature conventions	mt-	Myh6, Myh7, Tnnt2 for cardiomyocytes; Col1a1, Col3a1 for fibroblasts; Pecam1, Cdh5 for endothelial cells; Acta2, Myh11 for smooth muscle/pericytes; Csf1r, Cd68 for cardiac macrophages	The agent automatically switched to the mt- prefix, recognizing the organism from gene nomenclature conventions without explicit user specification; the agent automatically switched to heart-specific cardiac lineage markers

6. The psoriasis/GPR137 case study (Figure 6) is presented as experimentally validated, but the validation is a secondary analysis of existing experimental datasets (scCRISPRi and Perturb-seq), not new experiments. No new wet-lab validation was performed. Two additional points deserve more transparent treatment: (1) the human researcher selected the validation strategy (signature correlation analysis) from among 10 candidates proposed by the AI co-scientist; this human-in-the-loop step is essential but underemphasized, risking

overestimation of autonomous discovery capability; (2) the Spearman correlation for GPR137 is significant only under stimulated conditions (Stim8hr: rho = 0.61, p = 3.8e-3; Stim48hr: rho = 0.63, p = 4.7e-3) but not at rest (rho = 0.29, p = 0.21). The AI co-scientist did not predict this condition-dependence, and the paper does not discuss its biological implications. Whether the causal mechanism is activation-dependent rather than constitutive matters for the interpretation of GPR137's role in psoriasis.

For the ADHD case, while "identified" has been changed to "prioritized," the framing throughout (including Figure 5 caption: "revealing a plausible causal mechanism for ADHD risk") still reads as stronger than what the data support.

Response: We thank the reviewer for these points. We first explain why the GPR137 finding is scientifically interesting and why the validation is genuinely independent, and then address the condition-dependence observation.

Identifying the disease-relevant target gene at a GWAS locus is a long-standing open problems in human genetics, because a single regulatory variant typically affects multiple nearby genes, and disentangling them requires evidence beyond the original locus mapping. The scCRISPRi study reports that the rs887314 cis-regulatory element regulates three co-located candidates (GPR137, BAD, OTUB1) but does not resolve which of them mediates psoriasis risk[2]. Our AI co-scientist prioritized GPR137 by leveraging an independent Perturb-seq atlas of CD4+ T cell regulators, thereby generating a testable gene-level hypothesis that the source study did not provide. The AI co-scientist also proposed the ten candidate validation strategies; the human researcher's role was confined to selecting one of the agent-proposed strategies.

The agent's chosen strategy is itself a new cross-dataset integration approach. Rather than relying on a single readout from either study, the agent integrated the two complementary perturbation datasets by computing the Spearman correlation between (i) the gene-expression signature induced by CRISPRi knockdown of each candidate gene in the scCRISPRi screen and (ii) the perturbation signatures of CD4+ T cell regulators catalogued in the Perturb-seq atlas. This cross-screen, cross-modality signature-correlation procedure is not proposed in the source papers and is a flexible technique for transferring causal-gene prioritization across independent perturbation datasets at GWAS loci with multiple co-regulated candidates.

The validation is, moreover, genuinely independent. The Perturb-seq atlas used for validation was generated by a different laboratory using a different perturbation modality (Perturb-seq vs. scCRISPRi) and was released after the training-data cutoff of the LLMs underlying our agent, so its perturbation-effect estimates were not part of any pre-training corpus. The convergence between the agent's prioritized gene (GPR137) and the perturbation signatures in this held-out dataset, therefore, reflects out-of-distribution validation against new experimental data rather than retrieval of a memorized association.

Regarding the condition dependence of the GPR137 signal, the fact that the signal is significant only after stimulation but not at rest is consistent with the framing of both source datasets and

with the broader autoimmune-genetics literature. First, the Perturb-seq paper we used reports that "active regulators, and the gene programs they control change dramatically across stimulation conditions" and that roughly one third of $CD4^+$ T cell regulator clusters exert correlated effects in only one or two of the three (Rest, Stim8hr, Stim48hr) conditions[3]. A regulator detectable only after TCR engagement is therefore the expected case in this assay rather than an anomaly. Second, this pattern aligns with a broader feature of autoimmune disease genetics: a substantial fraction of $CD4^+$ T cell regulatory variants exert their largest effects only after activation, and these dynamic / stimulation-responsive effects are enriched at autoimmune GWAS loci[4]. Third, psoriasis pathology is driven by activated, IL-23/Th17-polarized $CD4^+$ T cells rather than resting cells, consistent with the cellular state in which the variant is most likely to exert disease-relevant effects[5]. We therefore describe GPR137's role at this locus as activation-dependent rather than constitutive, and frame the experimental support as concordant with the dataset's own framing of context-specific regulation.

For the ADHD case, we have further shifted the focus from causal identification to prioritization and the generation of plausible mechanisms. We also moved this case study to the Supplement. We further discuss the importance of human-in-the-loop for open-ended scientific tasks.

We have revised our manuscript to reflect those:

Revised Results section:

“The agent's proposed strategy is itself a novel data integration approach. Rather than relying on a single readout, the agent integrated the two complementary datasets by correlating CRE perturbation signatures (from scCRISPRi) with gene-knockdown signatures (from Perturb-seq). This cross-screen, cross-modality signature-correlation procedure is not proposed in either source paper and constitutes a generally applicable technique for transferring causal-gene prioritization across independent perturbation datasets at any GWAS locus with multiple co-regulated candidates.”

“Notably, the GPR137 knockdown effect reached significance only under stimulation (Stim8hr, Stim48hr) and not at rest (Spearman correlation = 0.29, p = 0.21), so we interpret GPR137's role at this locus as activation-dependent rather than constitutive. This pattern is consistent with the finding from the source Perturb-seq study that CD4+ T cell regulators vary substantially in their effects across stimulation conditions, and aligns with the established role of activated CD4+ T cells in psoriasis pathogenesis.”

“For each locus, the AI co-scientist prioritizes the top variant, links it to a candidate target gene, summarizes its molecular effects, and compiles a comprehensive markdown report detailing the functional consequences and biological significance of the prioritized gene.”

“We emphasize that open-ended scientific tasks such as hypothesis generation and mechanistic interpretation remain inherently human-in-the-loop. Paper2Agent generates and prioritizes candidate hypotheses, proposes validation strategies, and executes analyses at scale, while the

human researcher selects which hypotheses to pursue, chooses among proposed validation strategies, and judges biological plausibility. We therefore view Paper2Agent as a hypothesis-generation and analysis engine that augments human researchers, not an autonomous source of validated mechanistic claims."

7. The 74% success rate means 1 in 4 papers cannot be agentified at all. Among those that succeed, ~9% of queries produce incorrect answers. For scientific applications where correctness matters, this combined failure profile deserves more prominent discussion.

Response: Thank you for this comment. We expanded the Discussion to state that not all papers can currently be agentified, and that successful paper agents can still produce incorrect answers. We will also clarify appropriate usage scenarios: Paper2Agent is intended to improve access, reproducibility, and reuse, but novel scientific conclusions should still be independently verified.

We have revised our manuscript to reflect those:

Revised Results section:

"For these reasons, we view Paper2Agent as a tool for improving access, reproducibility, and reuse of published methods, rather than as an authoritative source for novel scientific conclusions. Users should inspect tool outputs, consult the accompanying provenance links and validation reports, and independently verify any novel scientific claim before acting on it."

8. The adversarial self-repair tests (36/36) cover a limited set of failure modes (path errors, missing dependencies, typos, deprecated APIs). Real-world repository degradation (breaking changes in upstream libraries, deprecated web APIs, data format evolution) was not tested.

Response: We clarify that the current stress tests focus on execution-level failures such as missing dependencies, path errors, typos, and deprecated APIs, and do not exhaustively test long-term repository drift, data format evolution, or external API changes. We have added new tests on additional repository degradations based on your suggestions. These include (i) breaking changes in upstream libraries: replacing `np.NaN`, `np.in1d` (removed in NumPy 2.0) and `pandas.DataFrame.append` (removed in pandas 2.0) in the tutorial notebooks; (ii) deprecated web APIs: redirecting the live GENCODE bucket to a non-existent legacy path and the AlphaGenome gRPC endpoint from `gdmscience.googleapis.com:443` to a sunset `alphagenome-preview.deepmind.googleapis.com:443`; and (iii) data format evolution: rewriting the inline VCF as strict VCFv4.2 with lowercase column headers, and rewinding the GENCODE annotation URL from `gencode.v46` to `gencode.v33` (a legacy schema lacking the `tag` and `mane_select` columns).

Paper2Agent's tutorial-executor sub-agent caught each failure from its runtime traceback and applied a minimal repair: for (i), `np.NaN`→`np.nan`, `np.in1d`→`np.isin`, and the `DataFrame.append` dead-code loop was removed; for (2), the dead GENCODE URL was

redirected to a locally pre-downloaded `gencode.v46` feather and the venv copy of `dna_client.py` line 897 was rewritten back to `gdmscience.googleapis.com:443` (verified by diffing the env-installed copy against the patched repo source); for (3), the VCF read was patched to `pd.read_csv(..., comment=None, header=3)` plus an explicit column rename to the uppercase names, and the GTF URL was reverted to `gencode.v46`. The patched notebooks then executed cleanly and produced fully functional MCP servers (18–22 tools each, comparable to the clean repo). Evaluated on the same 30-question AlphaGenome benchmark (15 tutorial + 15 novel) with Sonnet 4.6 + Paper2Agent MCP with 5 independent runs, (i) achieved 96.7% ± 1.9%, (ii) achieved 95.6% ± 2.2%, and (iii) achieved 96.7% ± 1.9%. Paper2Agent's self-repair, therefore, generalises beyond the original execution-level failures to the repository-drift modes the reviewer identified. Full details and per-fork diffs are added to the Supplementary Section "Adversarial Execution Evaluation".

We have revised our manuscript to reflect those:

Revised Results section:

"In adversarial repository-drift tests, Paper2Agent recovered functional MCP servers across injected dependency, file-path, typo, deprecated-API, deprecated-web-API, and data-format-evolution failures."

Revised Supplementary Note section:

"For each repository, we injected six categories of failures: (1) removal of required dependencies from environment specifications, (2) modification of input file paths to nonexistent locations, (3) insertion of typographical errors into code, (4) replacement of API calls with deprecated equivalents, (5) deprecated web APIs, and (6) data format evolution. In each case, the agent's iterative execution loop detected runtime failures from error tracebacks, applied targeted local fixes, and re-executed the workflow, producing functional MCP servers that reproduced the tutorial results. Evaluated on the 30-question AlphaGenome benchmark (15 tutorial + 15 novel) across 5 independent runs, the repaired servers achieved 95.6%–98.3% accuracy across all six failure categories (Supplementary Note)."

9. The non-biology evaluation (5 papers, 17 questions) remains thin for supporting a cross-domain generalization claim.

Response: Thank you for this comment. We note that the existing 74-paper / 300-question evaluation, although sourced from bioRxiv, already spans broader life and quantitative sciences beyond biology, including cheminformatics and drug discovery (e.g., medchem, PyrMol), clinical and translational medicine (e.g., TNBC target optimization, hepatocyte damage scoring), public health and epidemiology (e.g., PathogenSurveillance, antimicrobial-resistance surveillance), biostatistics (e.g., kernel machine regression, correlation-based clustering), biomedical NLP (e.g., PubMed hypothesis evaluation), and scientific image analysis (e.g., cryo-EM, cytokine arrays), across Python, R, Julia, Rust, and Java.

To further strengthen strictly non-life-science evidence, we built five additional Paper2Agent agents in non-biology domains: GenericML[6] for causal heterogeneous treatment effects, CausalImpact[7] for Bayesian structural time-series inference, Nashpy[8] for computational game theory, emcee[9] for affine-invariant MCMC in astrophysics, and conformal-selection[10] for FDR-controlled selective inference. We curated 25 novel-input benchmark tasks (five per paper) following the same protocol as the AlphaGenome novel query benchmark. Each task pairs a natural-language question with a small simulated dataset, and ground-truth values were computed by invoking the underlying R or Python packages directly. Across five independent runs (125 runs in total over the 25 tasks), Sonnet 4.6 + Paper2Agent MCP achieved 100.0 ± 0.0% accuracy against the ground truth computed through manual execution of the underlying packages.

We have revised our manuscript to reflect those:

Revised Results section:

"Across 42 execution-based tasks from 10 non-biology computational papers, Paper2Agent achieved 98.1% ± 0.8% accuracy across five independent runs, demonstrating generalization beyond computational biology."

Revised Supplementary Note section:

"We further extended our evaluation to include ten non-biology case studies spanning diverse scientific domains, including causal inference (grf), mechanistic interpretability (SAELens), natural language processing (Binoculars), computer vision (SAM2), tabular machine learning (TabPFN), heterogeneous treatment effects (GenericML), Bayesian time-series causal inference (CausalImpact), game theory (Nashpy), MCMC for astrophysics (emcee), and selective inference (conformal-selection). Across 42 execution-based benchmark tasks, Paper2Agent achieved 98.1% ± 0.8% accuracy across five independent runs, demonstrating generalization beyond computational biology."

Overall assessment:

The revised manuscript presents well-engineered work and the authors have been responsive to reviewer feedback with substantial new analyses and results. However, the central value proposition, that pre-computing MCP tools provides substantial benefit over giving a capable coding agent the repository directly, rests on a modest accuracy gap measured against an already-outdated baseline. The field of AI agents is moving extremely fast; what may have required a complex multi-agent pipeline in mid-2025 may be achievable by a general-purpose agent (like Opus 4.6 in Claude Code) in 2026. My assessment is that at this stage, the contribution is primarily one of engineering optimization (2.8x cheaper and 6.3x faster per query, 11 points more accurate) rather than a fundamental advance in how scientific knowledge is disseminated or used. I would recommend this work for a high-impact methods venue, where

the technical contributions would be appropriately valued, rather than Nature where the expectation is for broader conceptual advances.

Response: Thank you again for your helpful suggestions. While Paper2Agent delivers measurable engineering gains in accuracy and latency, it also has a fundamental conceptual contribution: it redefines how scientific knowledge is represented and used in the era of AI agents.

Reviewer #2 (Remarks to Authors);

The authors have provided a detailed and thoughtful response to the comments, and the updated manuscript is substantially strengthened by the addition of further empirical results and improved clarity. In particular, the expanded experimental evaluation demonstrating the generalizability of Paper2Agent across diverse types of papers, the analysis of failure modes, the inclusion of uncertainty measures, quantification of hallucination rates, and systematic ablations of system components all meaningfully enhance the rigor and impact of the work. The comparisons with existing executable artifact platforms such as CodeOcean are also valuable in this rapidly evolving space. The added statistical analyses appear appropriate, and the reported p-values support the stated conclusions. Overall, the major concerns raised in the initial review have been satisfactorily addressed.

Response: Thank you very much for your thoughtful feedback and suggestions; we really appreciate it! We are delighted that our revision has addressed your main concerns. We have carefully updated the paper to incorporate your additional suggestions.

A few points remain that could further improve the manuscript:

1. CodeOcean has recently introduced an AI agent platform with an MCP server (https://codeocean.com/blog/trusted-agents). Paper2Agent could be compared against this system.

Response: Thank you for this suggestion. We assessed the open-source Code Ocean MCP server and the Open Science Library directly. The MCP server exposes 26 tools, all of which are generic platform operations (`search_capsules`, `run_capsule`, `attach_data_assets`, etc.); none is a paper-specific scientific tool. Moreover, a Code Ocean capsule is created only when an author manually curates it, whereas Paper2Agent automatically ingests any public GitHub repository.

As an example, we tried to create an AlphaGenome MCP on Code Ocean but were unable to do so because Code Ocean didn't provide an API access. More broadly, authoring a Code Ocean capsule requires substantial manual effort from the original authors, including writing

orchestration scripts for end-to-end execution, creating and maintaining Docker environments, manually configuring dependencies, and satisfying Reproducible Run validation requirements. Paper2Agent removes this bottleneck by automatically generating MCPs without requiring human intervention.

To summarize, Code Ocean MCP depends on the proprietary Code Ocean platform and author-curated capsules, whereas Paper2Agent automatically generates open, paper-specific MCP tools directly from public GitHub repositories.

2. Although Paper2Agent outperforms previous baselines, its accuracy on benchmark questions is not perfect, implying that incorrect answers may occasionally be produced. This raises an important issue: how should users assess and trust responses to novel scientific questions, particularly in biological domains where errors could lead to incorrect conclusions? The authors should more explicitly caution readers about this limitation and clarify appropriate usage scenarios.

Response: Thank you for this suggestion. We have expanded the limitations section to caution that Paper2Agent outputs are not guaranteed to be correct. We explicitly state that users should inspect outputs, use provenance links and validation reports, and independently verify novel scientific conclusions.

We have added those to our revised paper.

Revised Discussion section:

"For these reasons, we view Paper2Agent as a tool for improving access, reproducibility, and reuse of published methods, rather than as an authoritative source for novel scientific conclusions. Users should inspect tool outputs, consult the accompanying provenance links and validation reports, and independently verify any novel scientific claim before acting on it."

3. The manuscript should clarify whether executable tutorials are required for the MCP tool to function. Additionally, can Paper2Agent automatically generate its own tutorials based on repository code?

Response: Thank you for this suggestion. Executable tutorials are not strictly required, and Paper2Agent can auto-generate its own tutorials from repository code and documentation.

Tutorials serve two distinct purposes in Paper2Agent: (i) they are the primary input from which the scanner identifies candidate tools, and (ii) they provide reference inputs and expected outputs that the test-and-repair loop uses to validate each extracted tool. When tutorials are present, they ground tool synthesis and verification in concrete, author-sanctioned workflows, which is why curated tutorials yield the highest accuracy. They are not, however, a hard prerequisite for producing a functional MCP.

We demonstrate this empirically with a new ablation on POP-TOOLS (a Python CLI toolkit for ML-assisted GWAS), in which the tutorials are removed from the source repository before invoking Paper2Agent. Paper2Agent first reads the README, the two CLI entry points, and the utility modules. Then it synthesizes four tutorial shell scripts covering quantitative-trait POP-GWAS, binary-trait POP-GWAS, single-variant POP-RARE, and gene-level burden POP-RARE on the bundled test data. The scanner identifies the two CLI entry points (`POP-GWAS.py`, `POP-RARE.py`), and the standard pipeline produces an MCP exposing four callable tools.

We then validated the resulting server against a human-executed reference across five analysis tasks, spanning both CLI tools (quantitative-trait GWAS, binary-trait GWAS, single-variant rare-variant test, gene-level burden test, and single-variant rare-variant test with sample-overlap correction). On every task, the output produced by invoking the MCP tool was byte-for-byte identical to the output of executing the underlying POP-TOOLS CLI command directly, so the auto-generated MCP reproduces the human-executed results exactly.

This shows that Paper2Agent can produce a functional MCP from a repository that ships no executable tutorials. Curated tutorials remain preferable when available because they encode source-specific conventions that documentation alone cannot capture, but they are not required to produce a working agent. This result is now reported in the Results section of the revised manuscript.

4. For the adversarial stress tests, it would strengthen the analysis to include the performance of functional MCP servers on the case studies and benchmark tasks.

Response: We thank the reviewer for this suggestion. We evaluated the four repaired MCPs for AlphaGenome on the same 15 tutorial and 15 novel benchmark questions used in the manuscript. The repaired servers answered 58 of 60 tutorial questions (96.7%) and 59 of 60 novel questions (98.3%) correctly, closely matching the curated-MCP baseline reported in the manuscript (98.7% tutorial, 100% novel).

In addition, we extended the adversarial stress tests to two further classes of repository drift: (i) deprecated web APIs, by redirecting the live GENCODE bucket to a non-existent legacy path and the AlphaGenome gRPC endpoint from `gdmscience.googleapis.com:443` to a sunset `alphagenome-preview.deepmind.googleapis.com:443`; and (ii) data format evolution, by rewriting the inline VCF as strict VCFv4.2 with lowercase column headers, and rewinding the GENCODE annotation URL from `gencode.v46` to `gencode.v33` (a legacy schema lacking the `tag` and `mane_select` columns). Paper2Agent's tutorial-executor sub-agent detected each failure from its runtime traceback and applied minimal repairs (e.g., redirecting URLs to a locally pre-downloaded feather, rewriting the venv-installed `dna_client.py` endpoint back, patching `pd.read_csv(..., comment=None, header=3)` with explicit column renames). Evaluated on the same 30-question AlphaGenome benchmark with Sonnet 4.6 + Paper2Agent MCP across 5 independent runs, the repaired servers achieved 95.6% ± 2.2% and 96.7% ± 1.9% on the two

new drift modes, statistically indistinguishable from the clean-MCP baseline. Combined with the original four failure categories (missing dependencies, broken file paths, typographical errors, deprecated API calls), the repaired servers achieved 95.6% to 98.3% accuracy across all six failure categories.

We have added those new results to our revised paper.

We have revised our manuscript to reflect those:

Revised Results section:

"Across 42 execution-based tasks from 10 non-biology computational papers, Paper2Agent achieved 98.1% ± 0.8% accuracy across five independent runs, demonstrating generalization beyond computational biology."

Revised Supplementary Note section:

"We further extended our evaluation to include ten non-biology case studies spanning diverse scientific domains, including causal inference (grf), mechanistic interpretability (SAELens), natural language processing (Binoculars), computer vision (SAM2), tabular machine learning (TabPFN), heterogeneous treatment effects (GenericML), Bayesian time-series causal inference (CausalImpact), game theory (Nashpy), MCMC for astrophysics (emcee), and selective inference (conformal-selection). Across 42 execution-based benchmark tasks, Paper2Agent achieved 98.1% ± 0.8% accuracy across five independent runs, demonstrating generalization beyond computational biology."

References

1. Miao, J. *et al.* Valid inference for machine learning-assisted genome-wide association studies. *Nat. Genet.* **56**, 2361–2369 (2024).
2. Ho, C.-H. *et al.* Genetic and epigenetic screens in primary human T cells link candidate causal autoimmune variants to T cell networks. *Nat. Genet.* **57**, 2536–2545 (2025).
3. Zhu, R. *et al.* Genome-scale perturb-seq in primary human CD4+ T cells maps context-specific regulators of T cell programs and human immune traits. *bioRxiv* 2025.12.23.696273 (2025).
4. Soskic, B. *et al.* Immune disease risk variants regulate gene expression dynamics during CD4+ T cell activation. *Nat. Genet.* **54**, 817–826 (2022).

5. Rendon, A. & Schäkel, K. Psoriasis pathogenesis and treatment. *Int. J. Mol. Sci.* **20**, 1475 (2019).
6. Chernozhukov, V., Demirer, M., Duflo, E. & Fernández-Val, I. Fisher–schultz lecture: Generic machine learning inference on heterogeneous treatment effects in randomized experiments, with an application to immunization in india. *Econometrica* **93**, 1121–1164 (2025).
7. Brodersen, K. H., Gallusser, F., Koehler, J., Remy, N. & Scott, S. L. Inferring causal impact using Bayesian structural time-series models. (2015).
8. Knight, V. & Campbell, J. Nashpy: A Python library for the computation of Nash equilibria. *J. Open Source Softw.* **3**, 904 (2018).
9. Foreman-Mackey, D., Hogg, D. W., Lang, D. & Goodman, J. emcee: the MCMC hammer. *Publ. Astron. Soc. Pac.* **125**, 306–312 (2013).
10. Jin, Y. & Candès, E. J. Model-free selective inference under covariate shift via weighted conformal p-values. *Biometrika* **113**, asaf066 (2026).

Editor's comments

1. Please reduce the length of the title to 75 characters (with spaces) or less, so that it fits on two lines in the final layout.

Response: We shortened the title to "Reimagining research papers as interactive and reliable AI agents," which contains 65 characters, including spaces.

2. Please provide the manuscript in .docx format. Currently it is in PDF format.

Response: We have provided the revised manuscript in .docx format.

3. Please add references to the abstract (if applicable).

Response: We have added references to the abstract.

4. Please reduce the Abstract to 230 words or less. Currently there are 263 words.

Response: We have reduced the abstract to 229 words.

5. Please create a separate reference list for any methods references, making sure that the numbering continues from the main text references.

Response: We have created a separate reference list for Methods references. The numbering continues from the main-text references, beginning with reference 43.

6. Please remove the main figures from the article file and re-supply them individually in an acceptable format such as EPS, AI, PS, PDF, PPT, PSD or XLS (for graphs) with editable vector files.

Response: We have removed all main figures from the article file and supplied them separately as editable PDF files.

7. You have more than one account on EJP. Please contact Nature Manuscripts (naturemanuscripts@nature.com) to confirm all accounts which belong to you, and your primary email address.

Response: We have contacted Nature Manuscripts at naturemanuscripts@nature.com and confirmed that the following accounts belong to the corresponding author: jcmiao@stanford.edu and jmiao24@wisc.edu. The primary email address is jcmiao@stanford.edu.

8. There are potential third party rights issues in the figures. Please check the sources of all illustrations and clarify whether permissions are needed to adapt or reproduce them. Please make sure to include the relevant details in third party rights table when you resubmit (more information below). If Biorender or a similar software has been used, please also ensure to

provide relevant licenses. In particular please check figure/s: main figure 1, 2a, 3a, 4a, extended figure 5a

Response: We have reviewed the sources of the illustrations in main Figures 1, 2a, 3a and 4a, and Extended Data Figure 5a. All illustrations were created by the authors and do not reproduce or adapt third-party material. No BioRender or similar software was used. The corresponding information has been included in the third-party rights table.

9. Please make sure to provide a third party rights table (more information below) when you resubmit. If Biorender or similar software has been used, please also ensure to provide relevant licenses.

Response: We have provided the completed third-party rights table with the resubmission.

10. Please provide a competing interest statement in the main text of the manuscript.

Response: We have included a Competing interests statement in the main manuscript.

11. Please provide an author contributions statement in the main text of the manuscript.

Response: We have included an Author contributions statement in the main manuscript.

12. Please remove the Extended data figures from the article file and re-supply them individually in EPS, JPEG or TIF format.

Response: We have removed all Extended Data figures from the article file and supplied them individually in JPEG format.

13. Please ensure that the text size in all figures is at least 5 pt Arial.

Response: We have verified that all figure text is set in Arial at a minimum size of 5 pt.

14. Funding section is provided along the acknowledgment section, please provide it as separate section.

Response: We have separated the Funding section from the Acknowledgements section.

Referee #1 (Remarks to the Author):

This revision adds updated Claude Opus 4.6 and Sonnet 4.6 baselines, expanded open-ended and non-biology benchmarks, and repository-drift stress tests. Some of my requests are handled well: the new Scanpy table convincingly shows the agent adapting markers to tissue and disease context. The GPR137 signal's restriction to stimulated cells is now reported and explained. The ADHD language is appropriately softened.

Unfortunately, one of my main concerns from last round is only partly resolved. I had argued that the accuracy gap reflected an outdated baseline that newer models would erase. On AlphaGenome they tested this directly, and an Opus 4.6 agent with repository access still trailed Paper2Agent. However, that test covered AlphaGenome alone; the large-scale benchmark that carries the central scalability claim (300 questions across the 74 agentified papers) still runs on the older Sonnet 4 model. So whether Paper2Agent's advantage persists as base models improve remains, in my view, open.

My other earlier concern also remains: the tools are built and validated on the same tutorials they are later tested on, and I would argue the new data recasts this as a question about what the benchmark scores actually measure. For the computational benchmarks, the tools are extracted from and validated against the same reference code that defines the ground truth, so these scores largely measure faithful reuse of one reference implementation. For a paper agent, that conformity is partly the point, but it means the results support reproducibility and reuse more directly than claims about scientific reasoning. An agent that writes a valid but different implementation could be deemed wrong unless equivalence is checked separately. Rather than a new experiment, I would value a subset audited for scientific equivalence, or a count of how many baseline answers were truly wrong rather than merely different.

Altogether, this leaves me unconvinced on significance. The engineering is genuinely strong: Paper2Agent converts a repository into a validated, queryable MCP server automatically, and repairs broken codebases. But the data substantiate the operational claims, eg improved reliability, reproducibility, and reuse of existing methods, more than the conceptual claim of a new paradigm for scientific communication. Establishing the latter (in my view needed for publication at Nature) would require isolating a capability attributable to agentification itself, rather than to the base model operating on the same repository, and the current experiments do not separate the two. The economics, including break-even and who maintains long-tail agents, are also not empirically resolved. My two remaining reservations are what the computational benchmarks actually measure and whether anything carries the significance claim beyond the engineering.

Response:

We sincerely thank the reviewer for the thoughtful feedback and for recognizing that our revision has addressed several requests. These suggestions have helped us further strengthen the manuscript.

First, we have now evaluated the Claude + Repo baseline using Sonnet 4.6 on the complete benchmark of 300 questions across 74 agentified papers. Sonnet 4.6 achieved 86.3% ± 1.06%, compared with 91.2% ± 1.6% for Paper2Agent with Sonnet 4 ($p < 0.0001$). Thus, there is still a significant performance benefit from using Paper2Agent even with newer models. We have added this result to the revised manuscript.

Second, we agree that agreement with a reference answer does not necessarily establish general analytical validity. This is particularly important for open-ended scientific analyses, for which multiple answers may be defensible, and the paper's reported result may not be the only reasonable conclusion. We have expanded the Discussion to clarify that single-reference benchmark scores primarily measure faithful execution of published methods. We also note that future evaluations of open-ended scientific tasks should examine the range of defensible answers rather than focus solely on agreement with a single reference.

Finally, we appreciate the reviewer's distinction between operational and broader conceptual claims. We frame Paper2Agent's principal contribution as converting static research artifacts into validated, provenance-linked, executable interfaces that improve the reliability, reproducibility, accessibility, and reuse of published methods. We do not interpret the current benchmarks as demonstrating general scientific reasoning or autonomous scientific discovery. We also agree that long-term maintenance and economic sustainability remain important next steps.

We have revised our manuscript to reflect those:

Revised Results section:

"Paper2Agent with Sonnet 4 achieved 91.2% ± 1.6% accuracy, outperforming Claude Code with direct repository access using Sonnet 4 (80.3% ± 2.3%; $p < 0.0001$) and Sonnet 4.6 (86.3% ± 1.06%; $p < 0.0001$)."

Revised Discussion section:

"For open-ended analyses, however, multiple answers may be defensible, so benchmarks based on agreement with a single reference should be interpreted primarily as measures of faithful execution rather than of analytical validity. Evaluating the range of defensible answers will be important for assessing AI systems on open-ended scientific tasks."

Referee #2 (Remarks to the Author):

The authors have provided detailed responses and further experimental results to the questions raised in the response to the revision. The additional updates to the manuscript have helped in improving the clarity, limitations, and robustness of Paper2Agent. I thank the reviewers for trying out to make the CodeOcean MCP server work. Also, it is good to mention that the outputs from Paper2Agent should not be used directly for scientific discovery without human verification since it can still make mistakes. This will help the readers understand the intended use of Paper2Agent. It is also great to see that Paper2Agent is robust to missing tutorials and

adversarial executions. For the adversarial stress test response, it seems like the authors have misplaced revised manuscript text in the response. Hopefully the results from the repaired MCPs are included in the revised manuscript. Overall, the remaining questions and concerns after the initial revision have been satisfactorily addressed.

Response:

We thank the reviewer for their thoughtful assessment and for recognizing the improvements in the manuscript's clarity, robustness, and discussion of appropriate use. We confirm that the results of the robustness experiments, including those involving omitted README files and shuffled tutorial code, are reported in the revised manuscript and are not limited to the response letter. We have also retained the explicit caution that Paper2Agent outputs should be independently validated before being used to support scientific conclusions.

We have revised our manuscript to reflect those:

Revised Results section:

"Paper2Agent also generated and validated a functional MCP from a repository without executable tutorials, demonstrating that curated tutorials are beneficial but not strictly required."

Revised Discussion section:

"Paper2Agent can generate hypotheses, propose validation strategies, and execute analyses at scale, but researchers remain responsible for selecting directions and evaluating evidence. We therefore view Paper2Agent as a tool for augmenting scientific discovery and improving access, reproducibility, and reuse of papers, rather than as an autonomous or authoritative source of scientific conclusions."