PaperDoctor: Evidence-Grounded and Actionable Feedback for Scientific Papers in Progress

Anonymous Author(s)
Affiliation
Address
email

Abstract

Autoresearch agents are reshaping the research pipeline, but they also let flawed claims enter the literature at scale. Human advisors catch such issues on in-progress drafts through careful, traceable feedback, yet advisor-style assess requires extensive manual effort and does not scale. To shift automated paper assessment from a judge to a diagnostician, we introduce PaperDoctor, an agent framework for pre-submission feedback with three key innovations: **(i) Holistic hierarchical pipeline.** Every paper is assessed across writing, layout, references, code, theory, prior work, and experiments through three layers: L1 paper-only screening runs cheaply on every submission; L2 typed verifiers route each claim to the skill that owns its evidence; and L3 reproducers rerun experiments by priority. **(ii) Evidence-grounded actionable feedback.** Each finding is a triple of an observation (Why), a pointer to a specific location such as a sentence, equation, or code line (Where), and a revision suggestion (How), making every critique auditable and actionable. **(iii) Effective experimental reproduction.** Beyond reading the paper, PaperDoctor selectively rebuilds and reruns experiments based on claim importance and compute budget, surfacing reproducibility gaps and quantitative limitations that are invisible from the manuscript alone. We evaluate PaperDoctor on human studies which are junior students on their pre-submission papers, yield 85% accuracy, notably high in claim assumption validation. We further evaluate PaperDoctor on 60 manuscripts across machine learning, natural and social-sciences, covering human- and agent-author papers with code. Overall, PaperDoctor produces more grounded feedback than human and LLM reviewers, its findings track paper quality, and its reproduction stage surfaces limitations missed in the paper's main body. To empower the community, we develop a demo interface that lets authors browse findings grounded on their paper. PaperDoctor reframes automated paper assessment as a diagnostic process rather than a verdict, taking a concrete step toward AI advisors that help authors raise the quality of scientific writing in the autoresearch era.

1 Introduction

"Don't find fault, find a remedy." — Henry Ford

Autoresearch agents that turn a proposed idea into a full manuscript that [22, 45, 28, 34, 2] increases the risk of producing work whose quality cannot be guaranteed, such as these claims are often difficulty to verify. Moreover, most human-written drafts are now at least partly AI-assisted, whether in writing, figure drawing, or literature survey. This trend raises paper volume while leaving draft quality uncontrolled. Existing automated reviewers [8, 18, 12, 13, 49, 32] do not close this gap. By returning a decision ("accept" or "reject") with a justification, they primarily serve to filter submissions, which is useful for the reviewer's side but uninformative for the author.

Human advisors catch exactly these issues during discussion with students [4, 31]. They go through the draft carefully: circling a claim with no experiments validation, underlining a theorem whose

Submitted to 40th Conference on Neural Information Processing Systems (NeurIPS 2026). Do not distribute.

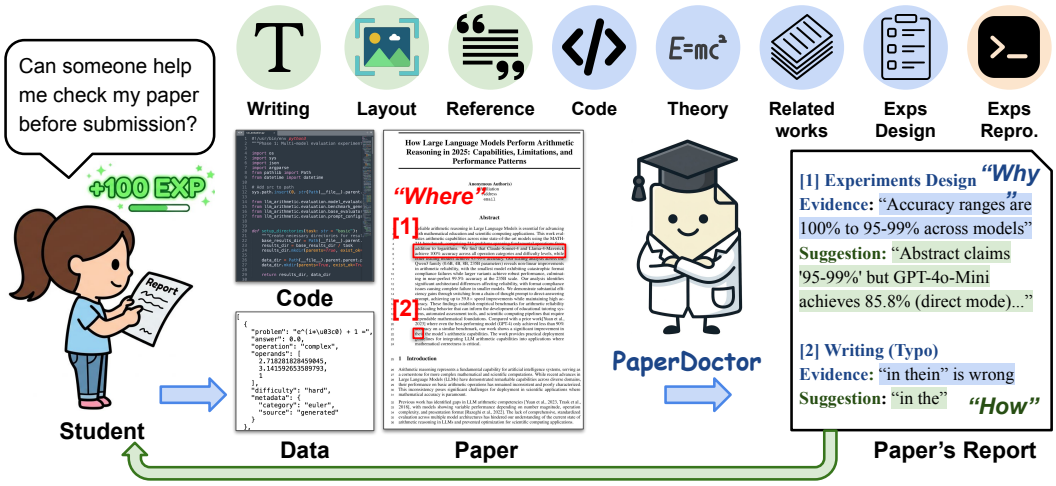


Figure 1: **PaperDoctor Agent provides evidence-grounded, actionable feedback.** Given a paper along with its code and datasets, PaperDoctor returns a holistic report of findings across multiple dimensions (writing, citations, figures, equations, code, reproduction, and related work). Each finding pairs an error type (*Why*) with a concrete suggestion (*How*) and is grounded (*Where*) in the paper itself, allowing authors to audit and learn from every critique.

assumption breaks, or flagging a figure that is unclear. Each mark on the page is a small diagnosis. As illustrated in Figure 1, it points to *where* the symptom is in the paper, explains *why* it is a problem, and prescribes *how* to fix it; in short, it is **evidence-grounded** and **actionable**. This is how students learn from an advisor's diagnosis on the page, which is fundamentally different from a reviewer's judgment. Yet such depth comes at a cost: it takes hours per paper, making it impossible to keep pace with agent-assisted writing [22, 45, 28, 34]. Current agents either write the paper for the author [34, 2] or grade it at the door [21, 27, 18, 33, 20]. Neither is what a student receives from an advisor: a careful diagnosis, like that of a doctor, that points to a specific section and says what to change. However, delivering this level of feedback is non-trivial. A paper is a multi-faceted artifact spanning presentation, implementation, experimental analysis, and more. This is why authors typically rely on feedback from a range of people, such as advisors, peers, and industry mentors, each catching issues the others miss.

Motivated by this, we ask: *can an agent provide diagnosis-level feedback at scale*? We introduce PaperDoctor, a research agent framework that delivers constructive feedback to human authors. Notably, PaperDoctor highlights the following: **(i) Holistic, hierarchical pipeline.** PaperDoctor decomposes a paper, together with its code and datasets, along dimensions ranging from writing to experimental reproduction, and organizes the assessment into three hierarchical levels of increasing cost and subjectivity. **L1 (surface screening)** runs cheaply on every submission and targets objective issues in writing, layout, references, and figures. **L2 (claim verification)** extracts atomic claims and routes each one to the skill that owns its evidence—web search for prior-work checks, a vision language model (VLM) for figure assessment, code analysis for implementation claims, and theory verifiers for derivations. **L3 (experimental reproduction)** selectively execute experiments, validating reproducibility and correctness at execution time. This design spends effort proportional to the cost of verification and keeps the full pipeline tractable. **(ii) Evidence-grounded, actionable feedback.** Each finding is a triple of an observation with a reason (Why), a pointer to a specific location in the paper (Where, *i.e.,* a sentence, equation, code line, or external URL), and a concrete suggestion for revision (How). This makes every critique both auditable and directly actionable for human authors. **(iii) Effective experimental reproduction.** Beyond reading the manuscript, PaperDoctor selectively prioritize and reruns experiments based on claim importance and compute budget. By executing code rather than only reading it, PaperDoctor surfaces reproducibility gaps and quantitative limitations that are invisible from the manuscript alone.

To validate the effectiveness of PaperDoctor, we first conduct a human study with junior student participants, collecting their in-progress paper drafts and asking them to rate the feedback produced by PaperDoctor. PaperDoctor reaches 85% agreement with their ratings. Moreover, we evaluate PaperDoctor on 60 manuscripts covering machine learning, the natural sciences, and the social sciences, including both human- and agent-written papers with accompanying code and data. This diverse benchmark allows us to systematically assess PaperDoctor's robustness across disciplines,

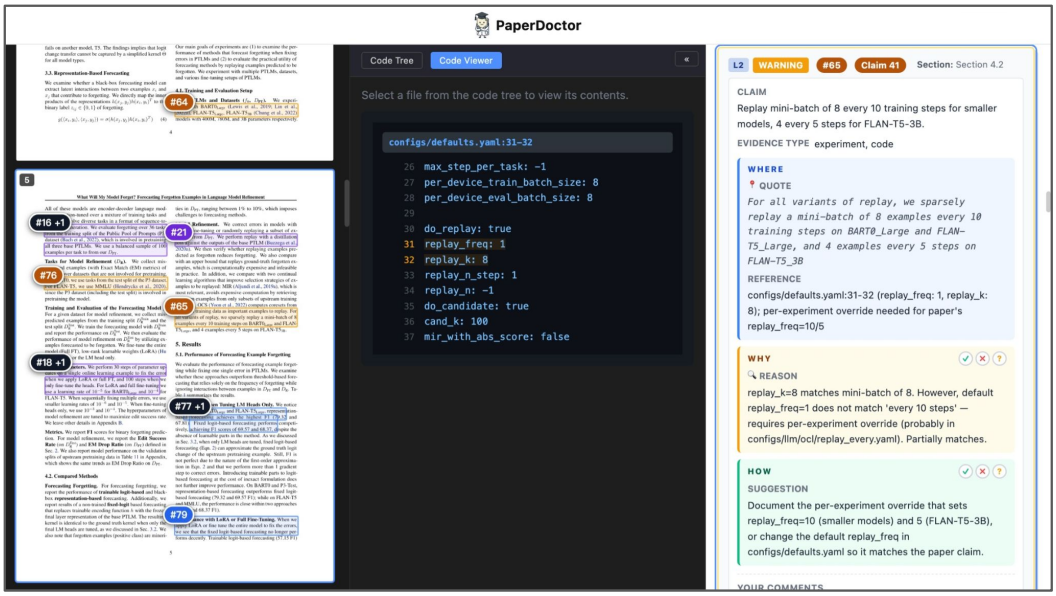


Figure 2: **The PaperDoctor Diagnosis Interface.** Authors upload their paper and code, and PaperDoctor returns the holistic report. **Left:** the paper, with each finding overlaid as a numbered, color-coded bubble anchored to the exact span it critiques (Where). **Middle:** the code viewer, so findings about implementation can be audited against the source. **Right:** the finding card for the selected highlight, showing the observation (Why) and a concrete suggestion for revision (How). In the example shown, PaperDoctor identifies an implementation-level mismatch: the paper claims a replay mini-batch of 8 every 10 training steps, but the default config (`configs/defaults.yaml:31-32`) sets `replay_freq=1`.

writing styles, and varying levels of methodological rigor. We found that (i) PaperDoctor produces more grounded feedback than human and LLM reviewers, its findings track paper quality, and its full coverage surfaces limitations missed from the paper main body. In particular, by executing the accompanying code and re-running key experiments, PaperDoctor uncovers discrepancies between reported and actual results that are otherwise difficult to detect through reading-only review.

Lastly, to empower the research community, we release a demo interface that lets authors browse findings anchored directly on their own papers (Fig. 2), making it easy to trace each critique back to its exact location.

2 PaperDoctor

2.1 Overview

Given a paper and its code, PaperDoctor aims to produce a set of findings $\{\mathcal{F}_i\}_{i=1}^{N}$, where each finding is defined as

$$\mathcal{F}_i = (f_i,\, e_i,\, s_i)\,. \tag{1}$$

Here, f_i is the finding itself, e_i is the *evidence* grounding it to a specific location (a sentence, equation, code line, or external URL), and s_i is a concrete *suggestion* for revision. The pair (e_i, s_i) lets the author audit and act on each finding without searching through the full paper. Notably, we distinguish findings into two severity levels. An *error* indicates that PaperDoctor is confident the issue is incorrect, while a *warning* indicates uncertainty and flags the finding for further clarification, such as a human check.

Paper-Code Parsing. Producing $\{\mathcal{F}_i\}$ by feeding the entire paper and codebase into a single model is infeasible: a paper is a long, multimodal document and a codebase is itself a large, structured artifact. Moreover, most downstream skills only need a targeted slice of the inputs (*i.e.,* reference verification needs only the bibliography; figure assessment needs only the rendered pages). We therefore run a single parsing step that produces three decomposed reusable artifacts:

$$\big(\mathcal{P}_t,\, \mathcal{P}_v,\, \mathcal{C}_t, \mathcal{B}\big) \leftarrow (\texttt{Paper},\, \texttt{Code}), \tag{2}$$

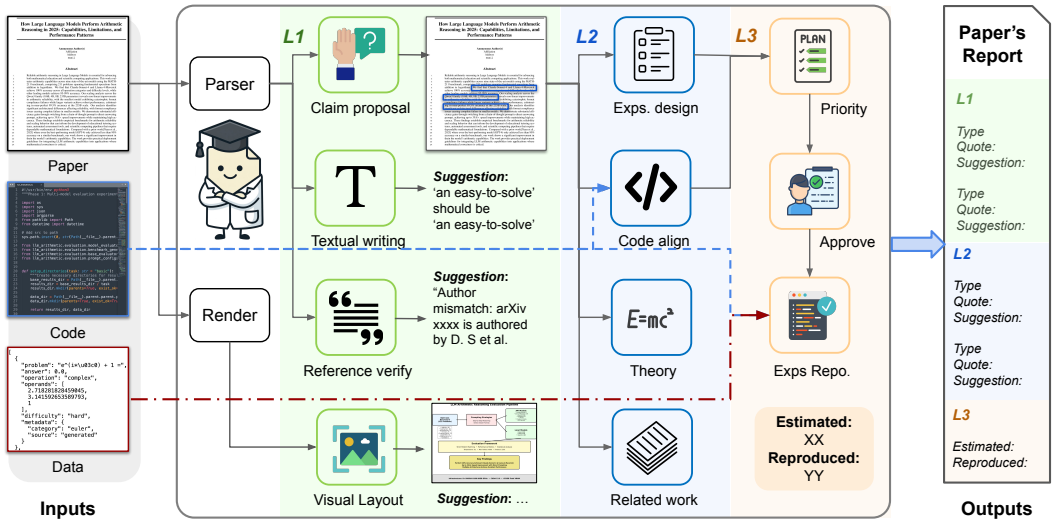


Figure 3: **Illustration of PaperDoctor pipeline.** A preparation step parses and renders the paper and code into a structured format and page images. **(i) The L1 surface-screening** stage runs four paper-only skills in parallel, including claim extraction. **(ii) The L2 claim-verification** stage dispatches each claim to the verifiers matching its evidence type and runs them in parallel. **(iii) The L3 experiment-reproduction** stage executes a prioritised, human-approved subset of the reproduction plan.

where $\mathcal{P}_t$ is the paper as section-organized markdown (via Mathpix[1]), $\mathcal{P}_v$ is the same paper rendered page-by-page as images for downstream Vision-Language Model (VLM) use, and $\mathcal{C}_t$ is the code indexed with tree-sitter[2] into per-file units. $\mathcal{B}$ denotes the parsed bibliography of the paper (*i.e.,* `.bib` file). Every downstream skill reads from this shared representation and requests only the section, page image, or code snippet it needs.

Hierarchical Pipeline. A paper spans many dimensions, and processing all of them in a single pass is infeasible. Different aspects demand different forms of evaluation, and these evaluations vary widely in cost: a writing check is a single LLM call, whereas experiment reproduction can consume hours of GPU execution. We therefore organize PaperDoctor as a hierarchical pipeline of three levels (L1–L3) that spends effort proportional to the cost of verification. As illustrated in Figure 3, L1 handles the most concrete, surface-level checks (such as presentation); L2 verifies individual claims along specific dimensions (such as theory or comparison with prior work); and L3 runs the most expensive stage, full experimental reproduction.

2.2 L1 – Surface Screening

This stage focuses on straightforward issues that can be easily addressed by browsing the paper.

Writing Review. We ask an LLM to read the paper section by section $\mathcal{P}_t$ and flag writing issues as it goes. Clear mistakes such as typos or grammatical errors are marked as *errors*, since they are unambiguously wrong. Stylistic issues, where the text is understandable but could be phrased more clearly, are marked as *suggestions* instead, leaving the final decision to the author. For every issue, we require the LLM to quote the original sentence verbatim as evidence, ensuring each finding can be traced back to a specific location in the paper.

L1: Writing

Evidence: Page 3 *"We trian the model on a large corpus of academic papers."*
Suggestion: There is a typo: "trian" should be "train".

Figure Review. A paper is as much a visual artifact as a textual one: its figures, tables, and overall layout are carefully curated by the authors and judged by readers at a glance. Yet most of this visual information is lost in markdown extraction. We therefore treat visual inspection as a separate check in PaperDoctor: we render the paper into page images $\mathcal{P}_v$ and ask a VLM to review them directly. Clear visual defects, such as figures overflowing the text margin or overlapping captions, are flagged as *errors*. More subjective issues, such as undersized fonts or insufficient color contrast, are flagged

[1] https://mathpix.com/
[2] https://github.com/tree-sitter/tree-sitter

as *warnings* for the author to judge. As with writing review, every finding must be grounded to a specific page or figure index as evidence.

L1: Figure

Evidence: Page 5, Figure 3 extends beyond the right text margin.
Suggestion: Rescale the figure width to `\linewidth`.

Citation Check. AI-assisted manuscripts routinely contain references that do not resolve to any real paper, and this is tedious to catch by reading the bibliography alone. Conditioned on the $\mathcal{B}$, we pair the agent with a web search backend: for each reference, the LLM issues up to three search queries and records a resolver URL only when the backend returns a genuine match, never synthesizing itself.

L1: Citation Check

Evidence: Reference [12] "Smith et al., Neural Reasoning in Transformers, NeurIPS 2023" returns no match.
Suggestion: The reference appears to be hallucinated. Please verify and replace with a valid source.

Claim Extraction. A paper makes dozens of arguments across its sections—novelty claims in the introduction, methodological choices in the method section, performance numbers in the experiments, and so on—and the value of PaperDoctor comes from checking each of them against its own evidence. A claim that is never extracted can never be verified. We therefore ask an LLM to densely extract every verifiable assertion the authors make, covering `[theory, code, experiments(designs), literature]`; a single claim may be tagged with multiple evidence types. Each claim is then dispatched to the corresponding L2 branches for verification.

L1: Claim Extraction

Claim: "Our method achieves 92.3% accuracy on ImageNet, outperforming prior state-of-the-art by 3.1 points."
Evidence: Introduction
Claim Type: `[Experiment, Related Work]`

Owning to the modular design, all four skills in L1 can run *in parallel*.

2.3 L2 – Claim Verification

In this stage, each claim by L1 claim extraction is sent to every verifier for further verification.

Code Verification. A common failure mode of AI-assisted drafts is that the described optimizer, architecture, or training setup does not match the released code. These mismatches are almost invisible to human reviewers, who rarely open the repo while reading. We therefore ask PaperDoctor to check each code-tagged claim directly against the source $(\mathcal{P}_t, \{\mathcal{C}_j\})$. For example, hyperparameter claims are often best verified by first inspecting configuration files before descending into the Python implementation: if the paper claims `AdamW` but the config specifies `Adam`, we flag a *warning*—the mismatch is real, but may be a stale config or a last-minute switch the author should confirm. If a component described in the paper is missing from the code altogether, we flag it as an *error*.

L2: Code Verification

Claim: "We train all models using the AdamW optimizer."
Evidence: `configs/train.yaml` line 14 specifies `optimizer: Adam`, which is conflict with the paper experiment settings
Suggestion: Mismatch between paper and code. Verify which optimizer was actually used.

Theory Verification. Errors in theoretical derivations are among the hardest to catch: a proof that reads smoothly often hides missing assumptions, skipped steps, or notation drift across equations, and even careful readers can miss these on a first pass. We therefore ask PaperDoctor to re-derive each argument in $\mathcal{P}_t$ step by step rather than summarize it. PaperDoctor jointly examines all theoretical content, including equations, variables, and the notation that links them across the paper, and records the full trace so that a superficial check is itself visible as a superficial trace. Specifically, PaperDoctor examines four aspects in turn: correctness of each step, hidden assumptions that the paper does not state, boundary or edge-case behavior, and notation consistency across derivations. A loss whose expectation silently drops between its definition and its final form is caught here, before it propagates into code or experiments.

L2: Theory Verification

Claim: "The expected loss reduces to $\mathbb{E}[\|x - \hat{x}\|^2]$ (Eq. 7)."

Evidence: Step from Eq. 6 to Eq. 7 drops the cross-term $\mathbb{E}[x^\top \hat{x}]$ without justification.

Suggestion: Missing assumption that x and $\hat{x}$ are uncorrelated. Please state explicitly or correct the derivation.

Literature Check. A common issue in scientific drafts is overstated novelty or weak engagement with the literature (*"no prior work addresses X"* when an earlier paper already does), and this bias is easier to catch by searching than by reading. Based on $(\mathcal{P}_t, \mathcal{B})$, PaperDoctor pairs an LLM with a web search backend to separate baseline comparisons, cited facts, and novelty assertions, so that each novelty assertion can be further labelled as `novel`, `incremental`, or `prior_art_exists`.

Note that this differs from the reference verifier in L1: the reference verifier only checks whether a cited paper exists and is correctly attributed, while this stage examines whether the surrounding literature *content* actually supports the paper's novelty and positioning claims.

L2: Literature Check

Claim: "No prior work addresses multi-modal reasoning over long video sequences."

Evidence: Web search returns Chen et al. (2023), "LongVid-Reasoner", which targets the same setting.

Suggestion: Novelty overstated. Relabel as prior_art_exists and cite Chen et al.

Experiment Design. While some claims can be closed-loop verified against external sources (literature) or formal content (theory, code), most claims in a paper rest on *experimental evidence* and crucially, whether the experiments themselves are well-designed determines whether the contribution is genuinely grounded. Experimental issues thus fall into two regimes: design mistakes (missing ablations, missing experiments for a stated contribution) that can be found from the paper alone, and reproduction mistakes that can only be found by running. This module handles the first, before any execution. Agent reviews whether each argument is supported by a corresponding experiment, along with fairness, ablation sufficiency, statistical rigour, baseline recency, and cherry-picking risk. It then emits a `reproduction-plan` entry per experiment listing the command, priority, feasibility, run mode (evaluation or training), and the numeric target lifted from the paper. No experiments run here; the plan is a declarative contract for L3.

L2: Experiment Design

Claim: "Our cross-modal attention module is the key component driving the gains over prior work."

Evidence: Table 3 reports only the full method versus the baseline; no ablation removes the cross-modal attention module to isolate its contribution.

Suggestion: Add an ablation that disables the cross-modal attention module and reports performance on the same benchmark.

Reproduction plan: `bash eval/ablate_attn.sh`, target $\Delta \geq 1.0$ point drop, priority high.

Notably, the L2 stage remains highly modular: all four verifiers, as well as the per-claim dispatch within each verifier, run *in parallel*.

2.4 L3 – Experiments Reproduction

After L1 and L2, PaperDoctor has already covered most aspects that can be assessed from the paper's main body. The remaining equally important is reproduction, which is challenging yet essential. Different papers come with very different experimental setups: some require only inference, others involve full training, and many depend on substantial resources such as GPU compute, datasets, and storage. We therefore design a separate L3 stage dedicated to reproduction.

Priority Ordering. It is worth noting that not every claim deserves an equal degree of attention. For example, an experiment that backs a main claim in the abstract carries far greater weight than a hyper-parameter sensitivity study, even though the latter may be cheaper to run. We rank experiments by their importance to the paper's central contributions and by expected *feasibility check*. PaperDoctor assigns each entry in the reproduction plan a priority label of `{high,medium,low}`. Under a compute budget, high-priority items run before medium and low, evaluations before training, and ready experiments before blocked ones.

Manual Approval. Reproduction consumes real compute and storage, and executes code that may affect the environment. A mis-typed claim could silently trigger a multi-hour training run. PaperDoctor therefore presents the L2 plan, annotated with estimated GPU hours, dataset size,

and storage footprint, for the author to approve. Only then does an LLM-driven executor handle environment setup, dispatch, and log parsing.

L3: Reproduction

Claim: "Our method achieves 78.4% accuracy on MMLU."

Evidence: Reproduced run yields 71.2% (`logs/mmlu_eval.log`), a 7.2-point gap.

Suggestion: Discrepancy exceeds the 1–2% tolerance. Flagged as *error*; verify evaluation protocol or reported number.

Report Results. Each executed experiment will receive a decision by comparing the reproduced value against the paper's reported one. Rather than imposing a fixed numeric threshold, PaperDoctor judges the verdict in context: it considers the metric type, the typical variance reported in the paper, and the magnitude of the original gap, and decides whether the result counts as a *pass*, a *warning* (partial match), or an *error* (OOM, divergence, or substantial mismatch). This gives the author a direct view of which claims hold up under execution and which diverge from what the paper reports.

3 Experiments

We design experiments to answer four research questions. ***Q1.*** How do human (such as students) perceive PaperDoctor's feedback? ***Q2.*** How does PaperDoctor perform across papers of varying sources and quality? ***Q3.*** How do PaperDoctor's findings compare to those of human and LLM reviewers? ***Q4.*** What unique findings does PaperDoctor uncover during experiment reproduction?

3.1 Settings

We conduct human studies with seven PhD students, collecting their pre-submission paper drafts. For the PaperDoctor generated report, we let these authors write and label every finding as *accepted*, *rejected*, or *uncertain*.

Furthermore, to study the performance across various paper types, we collect 60 papers from four diverse sources (as summarized in Tab. 1): twenty come from Agents4Science [2], half of them accepted as orals or spotlights and half rejected, so we could ablate AI-written outcome. Another twenty are from PaperBench [30]: human-written papers at top AI venues (ICML), all with released code. The remaining twenty are human-written papers from Nature Communications and Nature Human Behaviour (ten each), which pulls the evaluation outside machine learning and into the natural and social sciences.

Table 1: **Papers used in PaperDoctor evaluation.**

Author	Source	Domains	Size	Status	Code
AI	Agents4Science	AI & Science	20	Oral/Spot. & Reject	✓
Humans	PaperBench	AI (ICML)	20	Oral/Spotlight	✓
Humans	Nature Communications	Natural Science	10	Public	✓
Humans	Nature Human Behaviour	Social Science	10	Public	✓

We use `Claude-opus-4.7` as the base model and implement PaperDoctor as skills.

3.2 How do author assess PaperDoctor's feedback?

As shown in Fig. 5, we use PaperDoctor to generate 349 findings across 7 pre-submission papers, and ask each author to label every finding on their own paper. Authors accept 85% of the findings on average.

Where do authors agree most? As shown in Fig. 5, authors accept the majority of PaperDoctor's findings across all six dimensions, with acceptance rates ranging from 55% (figure) to 88% (theory) without rejection. L2 dimensions, which target specific claim types such as theory and related work, achieve consistently higher acceptance (77–88%) than L1 surface-level checks (55–74%), owing to their grounding in external evidence (*i.e.,* web search) and full-paper consistency over equations and notation. Among them, Experiment review yields the largest number of findings per paper, providing authors with a systematic reminder to revisit their experimental design.

Where do authors reject most? Figure dimension is the most rejected dimension, with nearly a third of findings marked as Rejected. Two reasons stand out: (i) current vision models still struggle to parse complex scientific figures and tend to hallucinate, and (ii) figure design is highly subjective, so automated judgments often diverge from what authors actually intend.

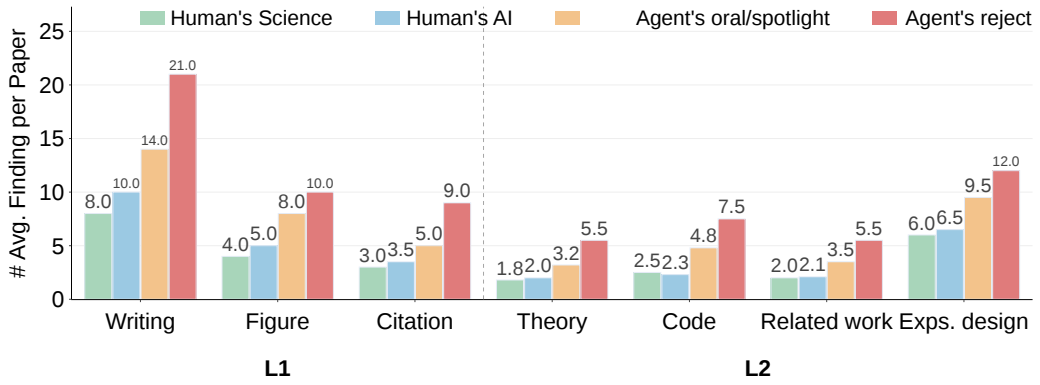


Figure 4: **Average findings per paper across paper categories.** We split papers into four groups by author and quality: human-authored science papers (Nature Communication, Nature Human Behavior), human-authored ML papers (PaperBench), and agent-written papers from Agents4Science, separated into oral/spotlight and rejected.

3.3 Are PaperDoctor's findings correlated with the paper category and quality?

Is PaperDoctor share the same role as Humans? Figure 4 breaks PaperDoctor's findings down by skill across four paper groups: human-written Nature papers, human-written AI papers from PaperBench, and agent-written papers that were either accepted (oral/spotlight) or rejected at Agents4Science. Several observations stand out.

Human-written papers have fewer suggestions than Agent's papers. Across every dimension, agent papers get more findings than human ones, suggesting PaperDoctor's finding counts track paper quality. Among human papers, Nature submissions sit at the low end on most dimensions, likely because their stricter review process catches surface issues earlier than open conference review. **Agent rejected papers have more suggestions than Agent top papers.** The gap is consistent across both L1 and L2 dimensions. Within L2, Experiment Design draws the most, ahead of theory, code, and related work. Many of these come from PaperDoctor checking the paper's internal self-consistency, for instance whether a number in the table matches the text, or whether the reported metric matches the one defined in the method.

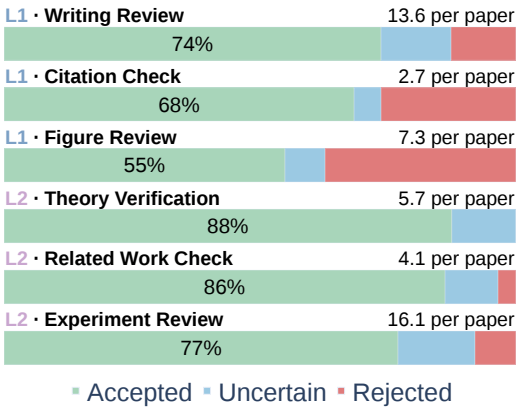


Figure 5: Author votings (Accepted / Uncertain / Rejected) on PaperDoctor's findings, broken down by individual modules.

Overall, PaperDoctor's finding counts align well with paper category and quality. Publicly accepted human papers carry the fewest issues, reflecting rounds of polishing, while agent-written papers, especially rejected ones, leave many issues a human collaborator would normally catch.

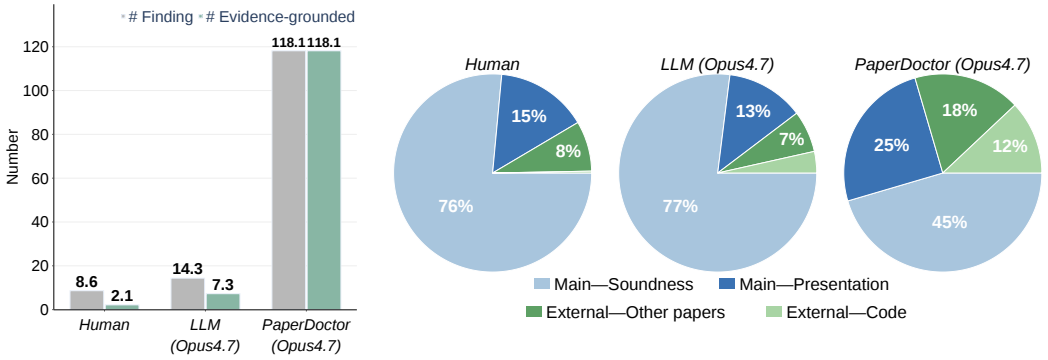
3.4 Comparison with Human Reviewers and LLM Reviewers

We study how human reviewers and LLM reviewers behave on the PaperBench papers [30] and obtain reviews directly from their authors. These reviews are not publicly available, ensuring that no current LLM has seen them during training.

How well do humans and LLMs ground their findings? We first ask whether human and LLM reviewers actually ground their findings in the paper, using an LLM judge to label each finding as evidence-grounded or not. We report results in Fig. 6a. We observe two gaps between baseline reviewers and PaperDoctor.

(i) Volume: Human reviewers produce only 8.6 findings per paper on average, and an LLM reviewer (Opus 4.7) with full access to the paper writes 14.3, far below PaperDoctor's 118.1. The gap is by design: rather than taking each argument for granted, PaperDoctor surfaces every claim that could potentially require verification.

(ii) Evidence grounding: Only 25% of human findings and 51% of LLM findings cite a concrete locus (a figure, table, equation, or quoted line), whereas PaperDoctor grounds every one of them (100%). These gaps are not accidental. For human reviewers, writing a precise locus during review is costly: it requires flipping through the paper, copying the exact span, and cross-checking line numbers, all competing with limited reviewing time. LLM reviewers fail differently: with the full paper in context, they treat every comment as self-evident thus omitting explicit citations that traiggered the issue.



(a) **Finding number** (with and without evidence grounding), compared across human reviewers, an LLM reviewer, and PaperDoctor.

(b) **Distribution of finding types.** *Main* refers to findings grounded in the paper body, *External* to those grounded in supplementary material, code, or other literature. *Soundness* covers method, theory, and experiments; *Presentation* covers writing and figures.

What do Humans, LLMs, and PaperDoctor focus on? In figure 6b, we classified the review content into four buckets: soundness and presentation within the paper body, and external checks against other literatures or against the released code. Human reviewers focus almost entirely on the paper body, with 76% of their findings on soundness and 15% on presentation; they rarely venture beyond the paper itself (8% on prior work, under 1% on code). The LLM reviewer (Opus 4.7) behaves almost identically, concentrating 90% of its attention on the paper body.

PaperDoctor distributes its attention very differently. Only 70% of its findings fall on the paper body, while 18% check other literature works and 12% verify the released code. The sharpest gap is on code: very small partition of human and LLM's touching, and 12% of PaperDoctor's. This shows that PaperDoctor plays a **complementary** role to human and LLM reviewers rather than a competing one: instead of producing a denser version of what they already write, PaperDoctor covers dimensions that a reader simply cannot reach.

Experimental Reproduction Analysis Due to the limited page space, please see Section D for a detailed reproduction analysis covering 60 papers, such as "*How does an L2 claim transformed into actual execution?*", "*Correlation between L2 priority and L3 reproduction success rate*", "*Correlation between reproduction mode and success rate?*", "*How does reproduction differ across paper sources?*".

4 Conclusions

We introduced PaperDoctor, an agent framework that shifts automated paper assessment from a judge to a diagnostician. Given a paper with the code, PaperDoctor produce each finding, which is a triple of an observation (Why), a pointer to a specific location (Where), and a concrete revision suggestion (How). PaperDoctor's findings are produced through a holistic hierarchical pipeline: L1 cheap surface checks run on paper surface, L2 typed verifiers route each claim to the skill that owns its evidence, and a L3 experimental-reproduction stage selectively reruns experiments under a priority budget. This pipeline scales effort to the cost of verification. Across a human study and diverse manuscripts spanning ML, the natural sciences, and the social sciences, PaperDoctor produces substantially more grounded feedback than human or LLM reviewers, its findings track paper quality, and its reproduction stage surfaces inconsistencies invisible from the manuscript alone. We release PaperDoctor together with a demo interface, in the hope of supporting more rigorous and reproducible research in the community.

References

[1] Kaito Baba, Chaoran Liu, Shuhei Kurita, and Akiyoshi Sannai. Prover agent: An agent-based framework for formal mathematical proofs. *arXiv preprint arXiv:2506.19923*, 2025.

[2] Federico Bianchi, Owen Queen, Nitya Thakkar, Eric Sun, and James Zou. Exploring the use of ai authors and reviewers at agents4science. *Nature Biotechnology*, 44(1):11–14, 2026.

[3] Bernd Bohnet, Vinh Q Tran, Pat Verga, Roee Aharoni, Daniel Andor, Livio Baldini Soares, Massimiliano Ciaramita, Jacob Eisenstein, Kuzman Ganchev, Jonathan Herzig, et al. Attributed question answering: Evaluation and modeling for attributed large language models. *arXiv preprint arXiv:2212.08037*, 2022.

[4] Gulfidan Can and Andrew Walker. A model for doctoral students' perceptions and attitudes toward written feedback for academic writing. *Research in Higher Education*, 52(5):508–536, 2011.

[5] Jun Shern Chan, Neil Chowdhury, Oliver Jaffe, James Aung, Dane Sherburn, Evan Mays, Giulio Starace, Kevin Liu, Leon Maksin, Tejal Patwardhan, et al. Mle-bench: Evaluating machine learning agents on machine learning engineering. *arXiv preprint arXiv:2410.07095*, 2024.

[6] Yuan Chang, Ziyue Li, Hengyuan Zhang, Yuanbo Kong, Yanru Wu, Hayden Kwok-Hay So, Zhijiang Guo, Liya Zhu, and Ngai Wong. Treereview: A dynamic tree of questions framework for deep and efficient llm-based scientific peer review. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 15662–15693, 2025.

[7] Ziru Chen, Shijie Chen, Yuting Ning, Qianheng Zhang, Boshi Wang, Botao Yu, Yifei Li, Zeyi Liao, Chen Wei, Zitong Lu, et al. Scienceagentbench: Toward rigorous assessment of language agents for data-driven scientific discovery. *arXiv preprint arXiv:2410.05080*, 2024.

[8] Mike D'Arcy, Tom Hope, Larry Birnbaum, and Doug Downey. MARG: Multi-agent review generation for scientific papers. In *arXiv preprint arXiv:2401.04259*, 2024.

[9] Christopher Foster. Openness in ai and downstream governance: A global value chain approach. *arXiv preprint arXiv:2509.10220*, 2025.

[10] Luyu Gao, Zhuyun Dai, Panupong Pasupat, Anthony Chen, Arun Tejasvi Chaganty, Yicheng Fan, Vincent Zhao, Ni Lao, Hongrae Lee, Da-Cheng Juan, et al. Rarr: Researching and revising what language models say, using language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 16477–16508, 2023.

[11] Tianyu Gao, Howard Yen, Jiatong Yu, and Danqi Chen. Enabling large language models to generate text with citations. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 6465–6488, 2023.

[12] Xian Gao, Jiacheng Ruan, Zongyun Zhang, Jingsheng Gao, Ting Liu, and Yuzhuo Fu. MMReview: A multidisciplinary and multimodal benchmark for LLM-based peer review automation. *arXiv preprint arXiv:2508.14146*, 2025.

[13] Zhaolin Gao, Kianté Brantley, and Thorsten Joachims. Reviewer2: Optimizing review generation through prompt generation. *arXiv preprint arXiv:2402.10886*, 2024.

[14] Mengze Hong, Di Jiang, Weiwei Zhao, Yawen Li, Yihang Wang, Xinyuan Luo, Yanjie Sun, and Chen Jason Zhang. Multimodal peer review simulation with actionable to-do recommendations for community-aware manuscript revisions. *arXiv preprint arXiv:2511.10902*, 2025.

[15] Tianyu Hua, Harper Hua, Violet Xiang, Benjamin Klieger, Sang Truong, Weixin Liang, Fan-Yun Sun, and Nick Haber. Researchcodebench: Benchmarking LLMs on implementing novel machine learning research code. *Advances in Neural Information Processing Systems*, 38, 2026.

[16] Qian Huang, Jian Vora, Percy Liang, and Jure Leskovec. Mlagentbench: Evaluating language agents on machine learning experimentation. *arXiv preprint arXiv:2310.03302*, 2023.

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

[18] Yiqiao Jin, Qinlin Zhao, Yiyang Wang, Hao Chen, Kaijie Zhu, Yijia Xiao, and Jindong Wang. Agentreview: Exploring peer review dynamics with llm agents. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 1208–1226, 2024.

[19] Rui Li, Jia-Chen Gu, Po-Nien Kung, Heming Xia, Xiangwen Kong, Zhifang Sui, Nanyun Peng, et al. Llm-reval: Can we trust llm reviewers yet? *arXiv preprint arXiv:2510.12367*, 2025.

[20] Shuaimin Li, Liyang Fan, Yufang Lin, Zeyang Li, Xian Wei, Shiwen Ni, Hamid Alinejad-Rokny, and Min Yang. Automatic paper reviewing with heterogeneous graph reasoning over llm-simulated reviewer-author debates. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 31717–31725, 2026.

[21] Weixin Liang, Yuhui Zhang, Hancheng Cao, Binglu Wang, Daisy Yi Ding, Xinyu Yang, Kailas Vodrahalli, Siyu He, Daniel Scott Smith, Yian Yin, et al. Can large language models provide useful feedback on research papers? a large-scale empirical analysis. *NEJM AI*, 1(8):AIoa2400196, 2024.

[22] Chris Lu, Cong Lu, Robert Tjarko Lange, Jakob Foerster, Jeff Clune, and David Ha. The ai scientist: Towards fully automated open-ended scientific discovery. *arXiv preprint arXiv:2408.06292*, 2024.

[23] MZ Naser. How llms cite and why it matters: A cross-model audit of reference fabrication in ai-assisted academic writing and methods to detect phantom citations. *arXiv preprint arXiv:2603.03299*, 2026.

[24] Azim Ospanov, Zijin Feng, Jiacheng Sun, Haoli Bai, Xin Shen, and Farzan Farnia. Hermes: Towards efficient and verifiable mathematical reasoning in llms. *arXiv preprint arXiv:2511.18760*, 2025.

[25] Abhilasha Ravichander, Shrusti Ghela, David Wadden, and Yejin Choi. Halogen: Fantastic llm hallucinations and where to find them. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1402–1425, 2025.

[26] ZZ Ren, Zhihong Shao, Junxiao Song, Huajian Xin, Haocheng Wang, Wanjia Zhao, Liyue Zhang, Zhe Fu, Qihao Zhu, Dejian Yang, et al. Deepseek-prover-v2: Advancing formal mathematical reasoning via reinforcement learning for subgoal decomposition. *arXiv preprint arXiv:2504.21801*, 2025.

[27] Pouria Rouzrokh, Bardia Khosravi, Parsa Rouzrokh, and Moein Shariatnia. Lattereview: a multi-agent framework for systematic review automation using large language models. *arXiv preprint arXiv:2501.05468*, 2025.

[28] Samuel Schmidgall, Yusheng Su, Ze Wang, Ximeng Sun, Jialian Wu, Xiaodong Yu, Jiang Liu, Zicheng Liu, and Emad Barsoum. Agent laboratory: Using LLM agents as research assistants. *arXiv preprint arXiv:2501.04227*, 2025.

[29] Zachary S Siegel, Sayash Kapoor, Nitya Nagdir, Benedikt Stroebl, and Arvind Narayanan. Corebench: Fostering the credibility of published research through a computational reproducibility agent benchmark. *arXiv preprint arXiv:2409.11363*, 2024.

[30] Giulio Starace, Oliver Jaffe, Dane Sherburn, James Aung, Jun Shern Chan, Leon Maksin, Rachel Dias, Evan Mays, Benjamin Kinsella, Wyatt Thompson, et al. Paperbench: Evaluating ai's ability to replicate ai research. *arXiv preprint arXiv:2504.01848*, 2025.

[31] Jacob Steiss, Tamara Tate, Steve Graham, Jazmin Cruz, Michael Hebert, Jiali Wang, Youngsun Moon, Waverly Tseng, Mark Warschauer, and Carol Booth Olson. Comparing the quality of human and chatgpt feedback of students' writing. *Learning and Instruction*, 91:101894, 2024.

[32] Pawin Taechoyotin and Daniel Acuna. Remor: Automated peer review generation with llm reasoning and multi-objective reinforcement learning. *arXiv preprint arXiv:2505.11718*, 2025.

[33] Cheng Tan, Dongxin Lyu, Siyuan Li, Zhangyang Gao, Jingxuan Wei, Siqi Ma, Zicheng Liu, and Stan Z Li. Peer review as a multi-turn and long-context dialogue with role-based interactions. *arXiv preprint arXiv:2406.05688*, 2024.

[34] Jiabin Tang, Lianghao Xia, Zhonghang Li, and Chao Huang. Ai-researcher: Autonomous scientific innovation. *arXiv preprint arXiv:2505.18705*, 2025.

[35] Xiangru Tang, Yuliang Liu, Zefan Cai, Yanjun Shao, Junjie Lu, Yichi Zhang, Zexuan Deng, Helan Hu, Kaikai An, Ruijun Huang, et al. Ml-bench: Evaluating large language models and agents for machine learning tasks on repository-level code. *arXiv preprint arXiv:2311.09835*, 2023.

[36] Nitya Thakkar, Mert Yuksekgonul, Jake Silberg, Animesh Garg, Nanyun Peng, Fei Sha, Rose Yu, Carl Vondrick, and James Zou. Can llm feedback enhance review quality? a randomized study of 20k reviews at iclr 2025. *arXiv preprint arXiv:2504.09737*, 2025.

[37] Nitya Thakkar, Mert Yuksekgonul, Jake Silberg, Animesh Garg, Nanyun Peng, Fei Sha, Rose Yu, Carl Vondrick, and James Zou. A large-scale randomized study of large language model feedback in peer review. *Nature Machine Intelligence*, pages 1–11, 2026.

[38] Haoxin Tu, Huan Zhao, Yahui Song, Mehtab Zafar, Ruijie Meng, and Abhik Roychoudhury. Agentic program verification. *arXiv preprint arXiv:2511.17330*, 2025.

[39] Sumanth Varambally, Thomas Voice, Yanchao Sun, Zhifeng Chen, Rose Yu, and Ke Ye. Hilbert: Recursively building formal proofs with informal reasoning. *arXiv preprint arXiv:2509.22819*, 2025.

[40] Yixuan Weng, Minjun Zhu, Guangsheng Bao, Hongbo Zhang, Jindong Wang, Yue Zhang, and Linyi Yang. Cycleresearcher: Improving automated research via automated review. *arXiv preprint arXiv:2411.00816*, 2024.

[41] Hjalmar Wijk, Tao Lin, Joel Becker, Sami Jawhar, Neev Parikh, Thomas Broadley, Lawrence Chan, Michael Chen, Josh Clymer, Jai Dhyani, et al. Re-bench: Evaluating frontier ai r&d capabilities of language model agents against human experts. *arXiv preprint arXiv:2411.15114*, 2024.

[42] Kevin Wu, Eric Wu, Kevin Wei, Angela Zhang, Allison Casasola, Teresa Nguyen, Sith Riantawan, Patricia Shi, Daniel Ho, and James Zou. An automated framework for assessing how well llms cite relevant medical references. *Nature Communications*, 16(1):3615, 2025.

[43] Yanzheng Xiang, Hanqi Yan, Shuyin Ouyang, Lin Gui, and Yulan He. SciReplicate-Bench: Benchmarking LLMs in agent-driven algorithmic reproduction from research papers. *arXiv preprint arXiv:2504.00255*, 2025.

[44] Zuyao Xu, Yuqi Qiu, Lu Sun, FaSheng Miao, Fubin Wu, Xinyi Wang, Xiang Li, Haozhe Lu, ZhengZe Zhang, Yuxin Hu, et al. Ghostcite: A large-scale analysis of citation validity in the age of large language models. *arXiv preprint arXiv:2602.06718*, 2026.

[45] Yutaro Yamada, Robert Tjarko Lange, Cong Lu, Shengran Hu, Chris Lu, Jakob Foerster, Jeff Clune, and David Ha. The ai scientist-v2: Workshop-level automated scientific discovery via agentic tree search. *arXiv preprint arXiv:2504.08066*, 2025.

[46] Shuo Yan, Ruochen Li, Ziming Luo, Zimu Wang, Daoyang Li, Liqiang Jing, Kaiyu He, Peilin Wu, George Michalopoulos, Yue Zhang, et al. Lmr-bench: Evaluating llm agent's ability on reproducing language modeling research, 2025. *URL https://arxiv. org/abs/2506.17335*.

[47] Zhengqing Yuan, Kaiwen Shi, Zheyuan Zhang, Lichao Sun, Nitesh V Chawla, and Yanfang Ye. Citeaudit: You cited it, but did you read it? a benchmark for verifying scientific references in the llm era. *arXiv preprint arXiv:2602.23452*, 2026.

[48] Xuanle Zhao, Zilin Sang, Yuxuan Li, Qi Shi, Weilun Zhao, Shuo Wang, Duzhen Zhang, Xu Han, Zhiyuan Liu, and Maosong Sun. Autoreproduce: Automatic ai experiment reproduction with paper lineage. *arXiv preprint arXiv:2505.20662*, 2025.

[49] Minjun Zhu, Yixuan Weng, Linyi Yang, and Yue Zhang. Deepreview: Improving llm-based paper review with human-like deep thinking process. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 29330–29355, 2025.

[50] Zhenzhen Zhuang, Jiandong Chen, Hongfeng Xu, Yuwen Jiang, and Jialiang Lin. Large language models for automated scholarly paper review: A survey. *Information Fusion*, 124:103332, 2025.

Table 2: **PaperDoctor vs. representative research agents**. **Reviewer Report**: produces reviewer-style prose, not only a score. **Grounded Evidence**: every finding is anchored to a concrete span/-figure/equation/code line. **Revision Suggestion**: output specifies *what to change*, not only *what is wrong*. **Text (LLM)**: reads body text, tables, and document structure. **Visual (VLM)**: reads figures as images. **Code Audit**: cross-checks paper claims against released code. **Exp. Reproduction**: re-executes experiments. **In-completed papers**: can the agent support or focus on these in-completed manuscripts. ✓=full, ✓=partial, ✗=absent.

System	Feedback Form			Assessment Coverage				Focusness
	Reviewer Report	Grounded Evidence	Revision Suggestion	Text (LLM)	Visual (VLM)	Code Imple.	Exp. Repro.	In-progress Papers
Autoresearch								
AI Scientist [45]	✓	✗	✗	✓	✗	✓	✗	✗
Peer Review								
MARG [8]	✓	✗	✓	✓	✗	✗	✗	✗
AgentReview [18]	✓	✗	✓	✓	✗	✗	✗	✗
Reviewer2 [13]	✓	✗	✓	✓	✗	✗	✗	✗
DeepReview [49]	✓	✗	✓	✓	✗	✗	✗	✗
TreeReview [6]	✓	✓	✓	✓	✗	✗	✗	✗
CycleReviewer [40]	✓	✗	✗	✓	✗	✗	✗	✗
MMReview [12]	✓	✓	✗	✓	✓	✗	✗	✗
Verification								
CiteAudit [47]	✗	✓	✗	✓	✗	✗	✗	✗
PaperBench [30]	✗	✓	✗	✓	✗	✓	✗	✗
AutoReproduce [48]	✓	✓	✗	✗	✗	✗	✓	✗
Feedbacks								
Review feedback [37]	✓	✓	✗	✓	✗	✗	✗	✗
Human advisor	✓	✓	✓	✓	✓	✓	✗	✓
PaperDoctor (ours)	✓	✓	✓	✓	✓	✓	✓	✓

A Related Work

Autoresearch for Papers End-to-end autoresearch pipelines chain ideation, experimentation, and drafting into a single agentic loop [22, 45, 28, 34, 2], increasingly supported by language and coding agents [15, 46, 43] that probe whether agents can implement and run published methods. Recent variants further explore tool-augmented, multi-agent collaboration, and long-horizon execution, pushing manuscripts toward greater autonomy. A shared trait, however, is that manuscripts are produced without an internal quality-control stage, leaving characteristic failure modes such as hallucinated citations, inflated novelty, mismatches between claims and the code that implements them, and unverified empirical statements—silently propagated into the final draft. PaperDoctor is complementary to this line of work: rather than producing papers, it consumes a draft together with its accompanying code and data, decomposes it across writing, references, theory, and experiments, and localises the artefacts that require revision before submission. In this sense, PaperDoctor acts as an internal advisor that closes the quality-control gap left open by current autoresearch agents.

Paper Peer Review A rapidly growing line casts LLMs as peer reviewers. Static prompting or fine-tuning yields a full review per paper [21, 8, 13, 49, 32, 27]; multi-agent variants simulate the review cycle with role-specialised agents [18, 33, 20]; decomposition-based TreeReview [6] recursively asks sub-questions; and CycleResearcher [40] closes the loop via iterative preference optimization. Multimodal and multidisciplinary variants [12, 14] read figures and tables; pre-submission assistance has been proposed as an ethically cleaner deployment [9]. At scale, the Review Feedback Agent [36] was deployed on 20k ICLR-2025 reviews. Parallel audits document persistent failure modes: prompt-injection susceptibility, sycophancy, and poor novelty calibration [19, 50]. These systems share a judge-oriented output contract, optimised against held-out reviewer opinions rather than author utility, and under-serve revision along three axes that PaperDoctor inverts. (i) Evidance-grounded. Verdicts are rarely tied to a specific sentence, equation, or code line; PaperDoctor emits (finding, evidence, suggestion) triples attached to concrete artefacts. (ii) Claim-level auditability. Holistic judgements let

individual unverifiable assertions pass silently; PaperDoctor extracts claims and verifies each one. (iii) Cost tiering. Review pipelines are binary: every check always runs or never runs; PaperDoctor exposes a three-tier pipeline so cheap screens are default, typed verifiers are routed by evidence, and full reproduction is gated on explicit author approval.

Paper Verification For rigorous assessment, a paper can be treated as a verifiable system whose claims must be checked along multiple dimensions. (i) Citation verification. LLM drafts routinely contain fabricated references, with audits reporting hallucination rates significantly [44, 23]; dedicated verifiers [47, 42] and attributed-generation frameworks [11, 10, 3, 25] supply the primitives for PaperDoctor's reference verifier. (ii) Theoretical-claim verification. LLM-based provers [26, 1, 39, 24] and agentic program verifiers [38] combine informal reasoning with Lean and Coq checking. PaperDoctor incorporates this perspective: it densely extract informal argument and check whether it matches formal or executable content. Moreover, the reproducbility is considered (iii) Experiment reproduction. Beyond textual verification, reproducibility benchmarks such as [5, 16, 35] evaluate ML engineering on Kaggle- and repo-scale tasks, while [17, 29, 7, 41] extend this to software, computational, scientific, and frontier-R&D settings, with top agents still below 40% execution accuracy [15, 46, 43]. PaperDoctor internalises the lesson that full reproduction is expensive and noisy: L3 issues a *prioritised* plan from the paper's own claims and executes only with author approval, spending compute where evidence, not bandwidth, is the constraint.

B Experiment Details

Implementation. PaperDoctor is implemented as a set of Claude-skills orchestrated by the tiered pipeline of Section 2. L1 and L2 skills use CLAUDE-OPUS-4.7 for text reasoning and the same model with vision enabled for `read-vis`. Web lookups in `read-bib` and `read-prior` use the built-in WebSearch backend, bounded to at most three queries per claim. L3 reproduction runs on a single node with an NVIDIA A100 (80 GB). All 60 papers go through the same pipeline with the same skill configuration; the L3 reproduction gate is enabled uniformly so that `read-exp` plans are dispatched to `run-exp` under a fixed 30-minute per-claim budget.

C Pipeline Algorithm

Algorithm 1 gives the full pseudocode for the three-tier PaperDoctor pipeline summarised in Section 2.1. Stage 0 (PREPARE) is executed once per paper; L1 and L2 skills run in parallel over the shared artefact bundle $\mathcal{A}$; the L3 RUNEXP stage is only executed when the user explicitly approves the reproduction plan produced by `read-exp`.

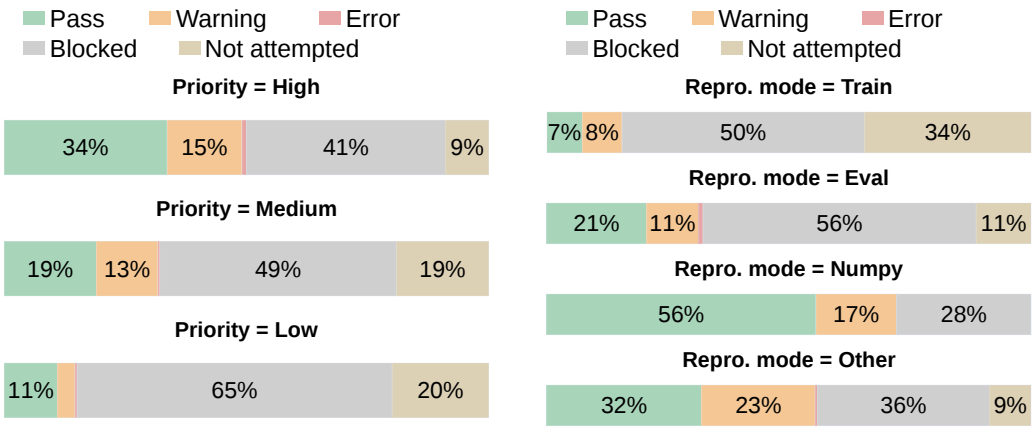
Algorithm 1 PaperDoctor review pipeline

Require: paper $\mathcal{P}$, code $\mathcal{C}$, data $\mathcal{D}$
$\mathcal{A} \leftarrow \textsc{Prepare}(\mathcal{P}, \mathcal{C})$
$\{\mathcal{Q}, \mathcal{F}_{\text{txt}}, \mathcal{F}_{\text{vis}}, \mathcal{F}_{\text{bib}}\} \leftarrow \text{L1}(\mathcal{A})$ *// parallel*
$\mathcal{F}_{\text{L2}} \leftarrow \emptyset$
for all $c_i \in \mathcal{Q}$ **in parallel do**
  for all $t \in T_i$ **do**
    $\mathcal{F}_{\text{L2}} \leftarrow \mathcal{F}_{\text{L2}} \cup \{\textsc{Verify}_t(c_i, \mathcal{A})\}$
  end for
end for
if user approves reproduction **then**
  $\mathcal{F}_{\text{L3}} \leftarrow \textsc{RunExp}(\mathcal{F}_{\text{L2}}^{\text{exp}}, \mathcal{C}, \mathcal{D})$
else
  $\mathcal{F}_{\text{L3}} \leftarrow \emptyset$
end if
return $\textsc{Report}(\mathcal{P}, \mathcal{F}_{\text{txt}} \cup \mathcal{F}_{\text{vis}} \cup \mathcal{F}_{\text{bib}} \cup \mathcal{F}_{\text{L2}} \cup \mathcal{F}_{\text{L3}})$

D Reproduction Analysis

How does an L2 claim transformed into actual execution? Figure 8 shows the end-to-end funnel of L2 experiments claims forwarded to L3. PaperDoctor surfaces 32.0 experimental claims per paper. About half (45.8%) are routed to L3 with a reproduction plan, while the rest are flagged as not requiring re-execution, for example simple inference checks or claims already self-verified within the paper. Of the routed claims, only 1.95 (6.1% of all L2 claims) clear the feasibility check and are ready to run, while 12.7 (39.8%) are blocked, most often by missing model weights, data access, or heavy GPU compute requirements. After human approval, 6.1 are eventually attempted, leading to 3.6 passes, 2.4 warnings per paper. End-to-end, only 11.3% of L2 claims yield a reproduced number, with conditional on execution the success rate (pass plus warning) is 59.5%.

Correlation between L2 priority and L3 reproduction success rate? Figure 7a breaks down L3 outcomes by L2-assigned priority. Pass rates drop monotonically from 34% for High-priority claims to 19% for Medium and 11% for Low, validating L2's prioritisation. Higher-priority claims tend to come with richer reproduction specifications, including explicit numeric targets, well-described commands, and clearly identified entry points, and therefore more often clear the feasibility check. Lower-priority claims, by contrast, are typically auxiliary numbers such as parameter sweeps or sensitivity studies, for which authors often do not release a runnable script in the first place. As a result, most are blocked rather than executed.



(a) By L2-assigned priority.

(b) By reproduction mode.

Figure 7: **L3 reproduction breakdowns**. **(a)** Pass rate by L2 priority. **(b)** Pass rate by reproduction mode.

Correlation between reproduction mode and success rate? Figure 7b breaks the same outcomes down by reproduction mode. Pass rates span an order of magnitude: 56% for Numpy-style standalone scripts, 32% for Other (Mainly from the social science papers), 21% for Eval (running released checkpoints), and only 7% for Train. **Training is by far the hardest mode**, with 84% of plans blocked before any code runs, because training at once needs jointly the right environment, the data, and serious compute.

How does reproduction differ across paper sources? Figure 9a breaks reproduction down across five paper sources. Plan density varies by an order of magnitude, from 24.9 plans per paper for Social-sci down to 4.6 and 2.5 for Agent4Science accepted and rejected, reflecting how much of each domain's claims map to concrete numerical targets: **social-science papers are dominated by statistical tests**, while **agent-written papers contain very few experiments** overall and are often short-paper format. In Fig. 9b, conditional pass rates (defined as Pass / Attempted) tell a different story: Nature science leads at 67.6%, while Agent-rejected drops to 12.5%. Two failure modes are visible. PaperBench struggles before execution, with many plans blocked by environment or compute issues, but the experiments that do run usually match. Agent-rejected struggles after execution: most of its plans run, but the numbers rarely line up with what the paper reports. Notably, **"Pass / Attempted"** measures how often a paper's experiments hold up *once execution is feasible*. By this metric, peer-reviewed papers from established venues (Nature, PaperBench oral/spotlight) sit at the

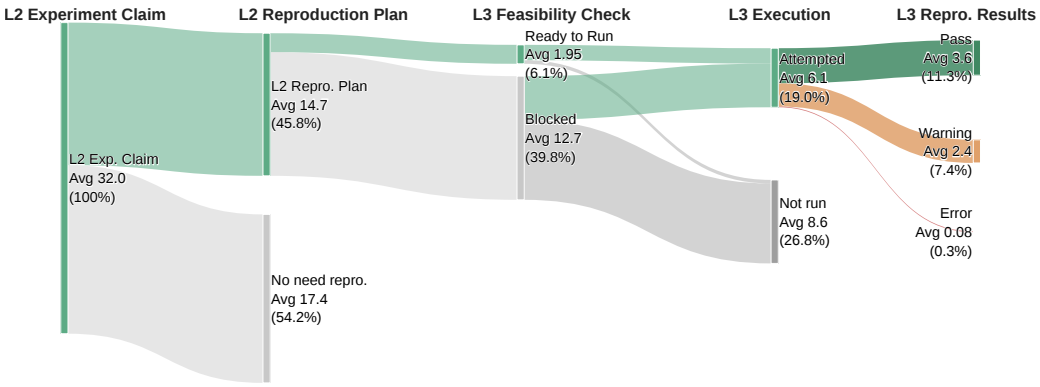
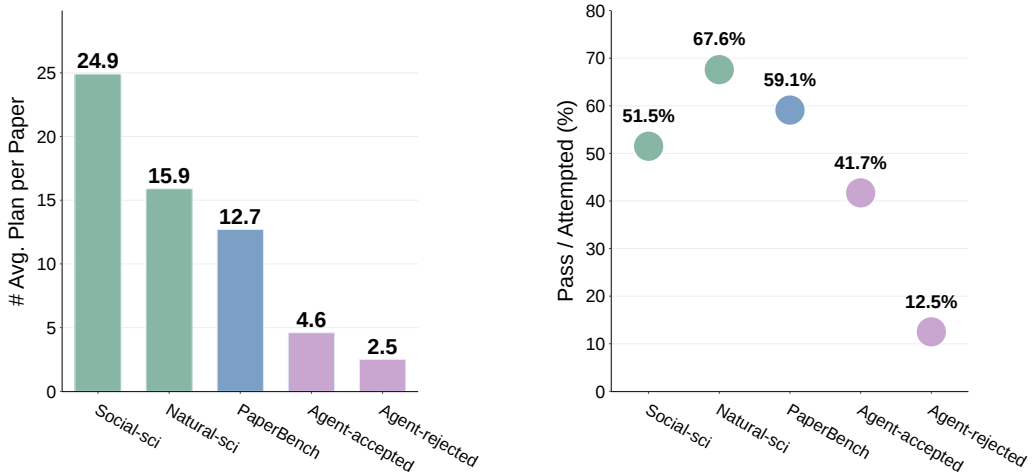


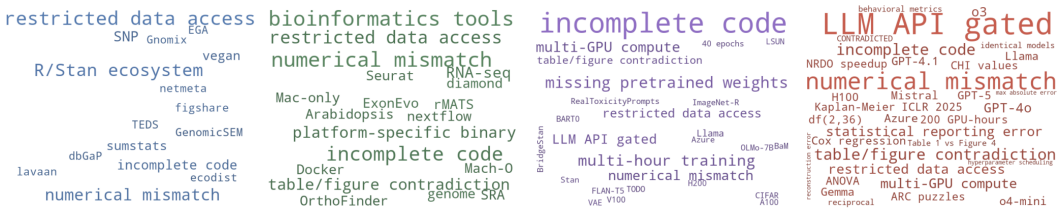
Figure 8: **End-to-end reproduction funnel across 60 papers.** PaperDoctor surfaces 32.0 experimental claims per paper on average; 14.7 receive a reproduction plan, 1.95 clear the feasibility check, and 6.1 ultimately execute. Of those that run, 3.6 pass and 2.4 yield warnings. The end-to-end yield is 11.3% reproduced claims per paper, or 59.5% conditional on execution.

top, while agent-written rejects sit at the bottom, suggesting that conditional pass rate can serve as a useful indicator of empirical reliability.



(a) Avg. L3 plans number. (b) Conditional pass rate.

Figure 9: **L3 reproduction breakdowns by paper source. (a)** Average L3 plans per paper. **(b)** Conditional pass rate (pass/attempted).



(a) Social Science (b) Natural Science (c) PaperBench (AI) (d) Agent4Science

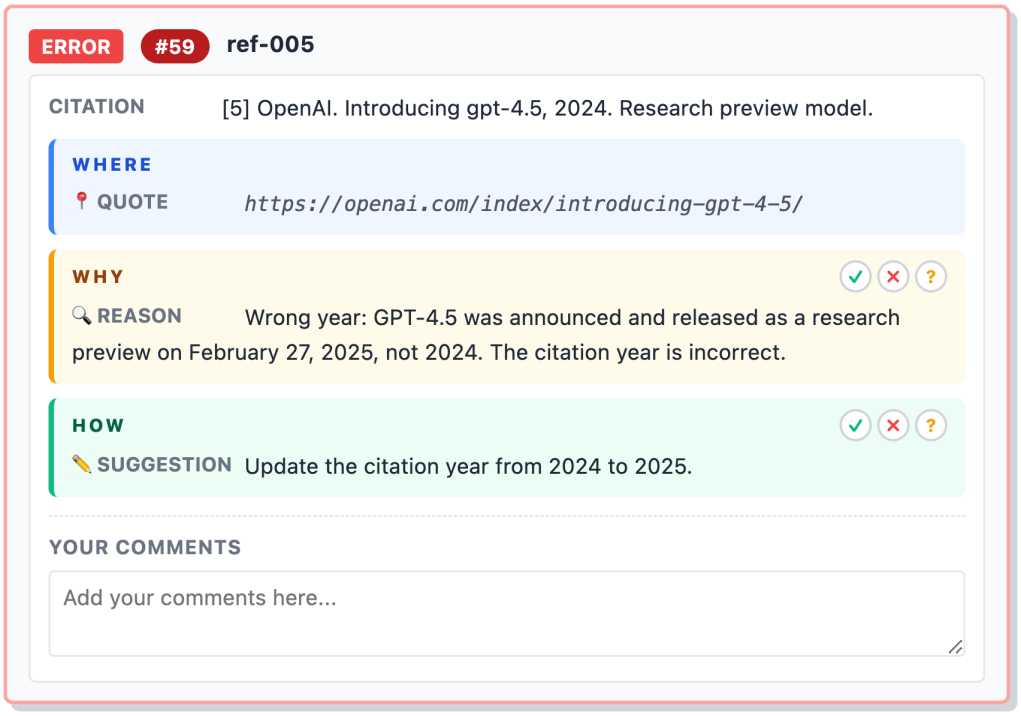
Figure 10: **Distribution of reasons that why PaperDoctor fail at reproduction stage.**

What kinds of failures dominate each domain? Figure 10 shows the failure reasons by paper source, and the patterns are very different. Social Science is mostly stuck on restricted-access data and R/Stan tooling. Natural Science runs into platform-specific binaries and bioinformatics pipelines. PaperBench is dominated by infrastructure cost: missing weights, gated APIs, and multi-GPU training. Agent4Science fails in a different way altogether: its training environment is not hard to satisfy, but the numbers disagree with what the paper reports. Together, these patterns show that feasibility remains a major bottleneck for reproduction, and underline the need for more reproducible and openly documented research artifacts.

E Case Studies

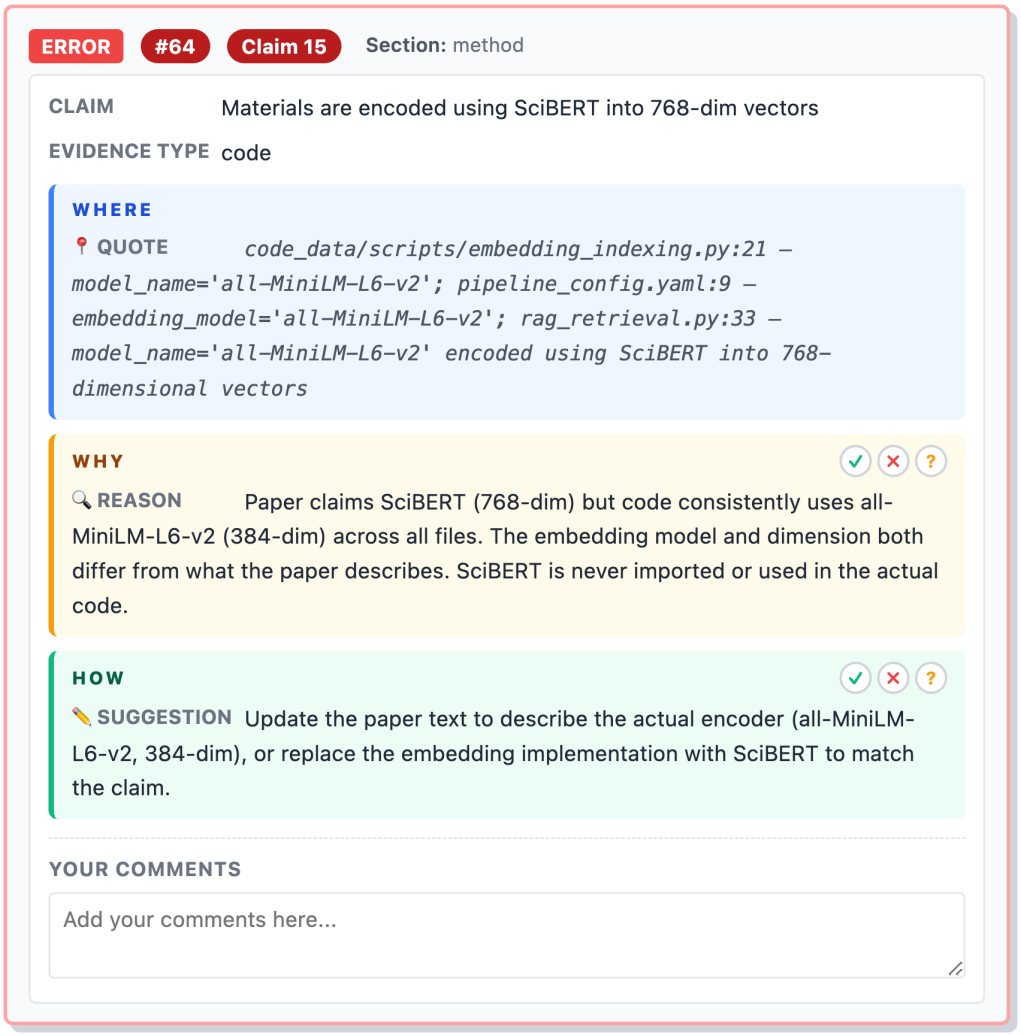
The aggregate numbers in Section 3 describe PaperDoctor's behaviour at scale, but they do not show what an individual finding looks like to an author opening the report. We walk through five representative cases drawn from our 60-paper benchmark, one for an L1 surface skill, three for distinct L2 typed verifiers, and one for an L3 experimental reproduction. Each case follows the same triple of *Why*, *Where*, and *How*, and was selected because the issue would be invisible (or at least very hard to spot) under conventional reading-only review.

L1 Citation Check: a fabricated future date.



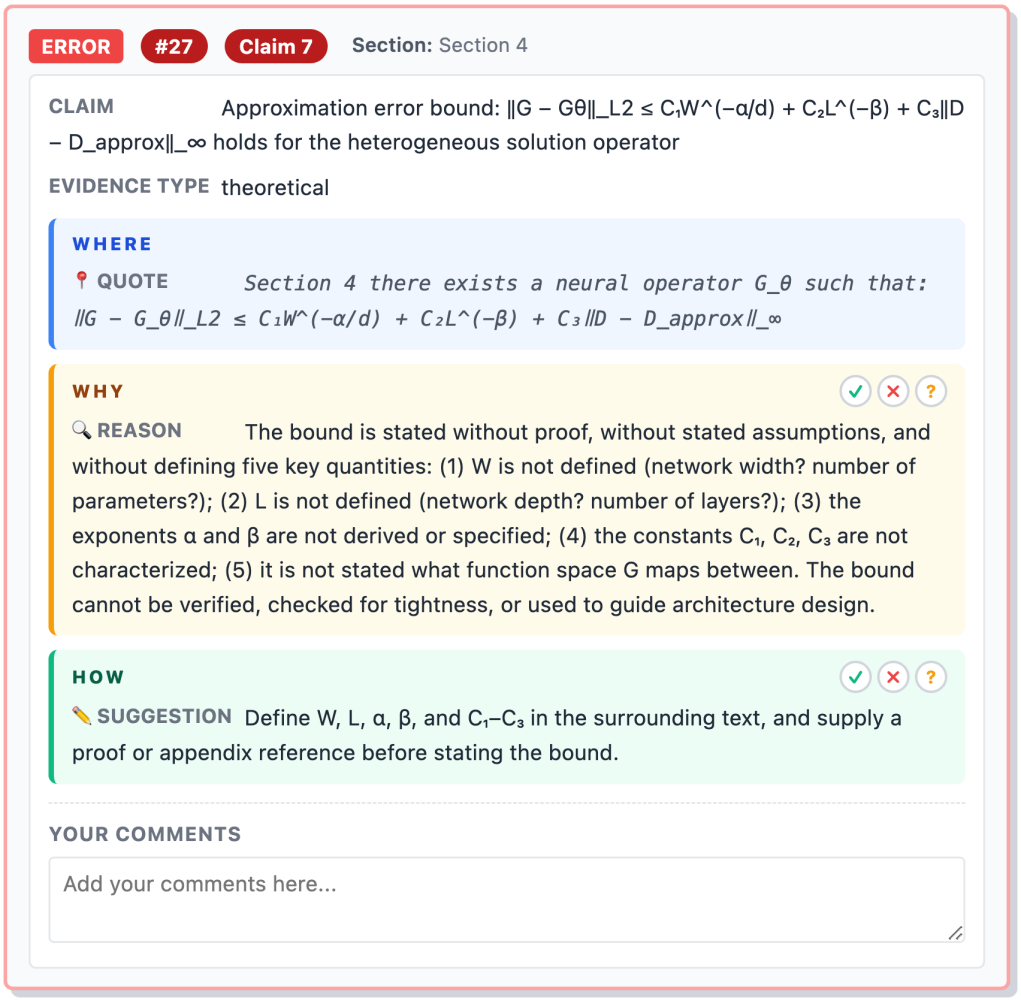
In *Reasoning Models Outperform Standard Language Models in De Novo Protein Design* (Agents4Science accepted), the bibliography contains the entry "[5] OpenAI. Introducing GPT-4.5, 2024. Research preview model." PaperDoctor's citation verifier issues a web search for this reference and finds that *GPT-4.5* was announced and released on February 27, 2025, not in 2024. Although the manuscript itself appeared in late 2024, the cited release date is therefore chronologically impossible. Standard citation checkers accept any correctly-formatted entry; only a search-grounded verifier flags this kind of temporally inconsistent metadata, which is a recurring failure mode of AI-assisted drafts that synthesise plausible-sounding venues and years without grounding them in real release notes.

L2 Code Verification: paper claims an encoder its code never instantiates.



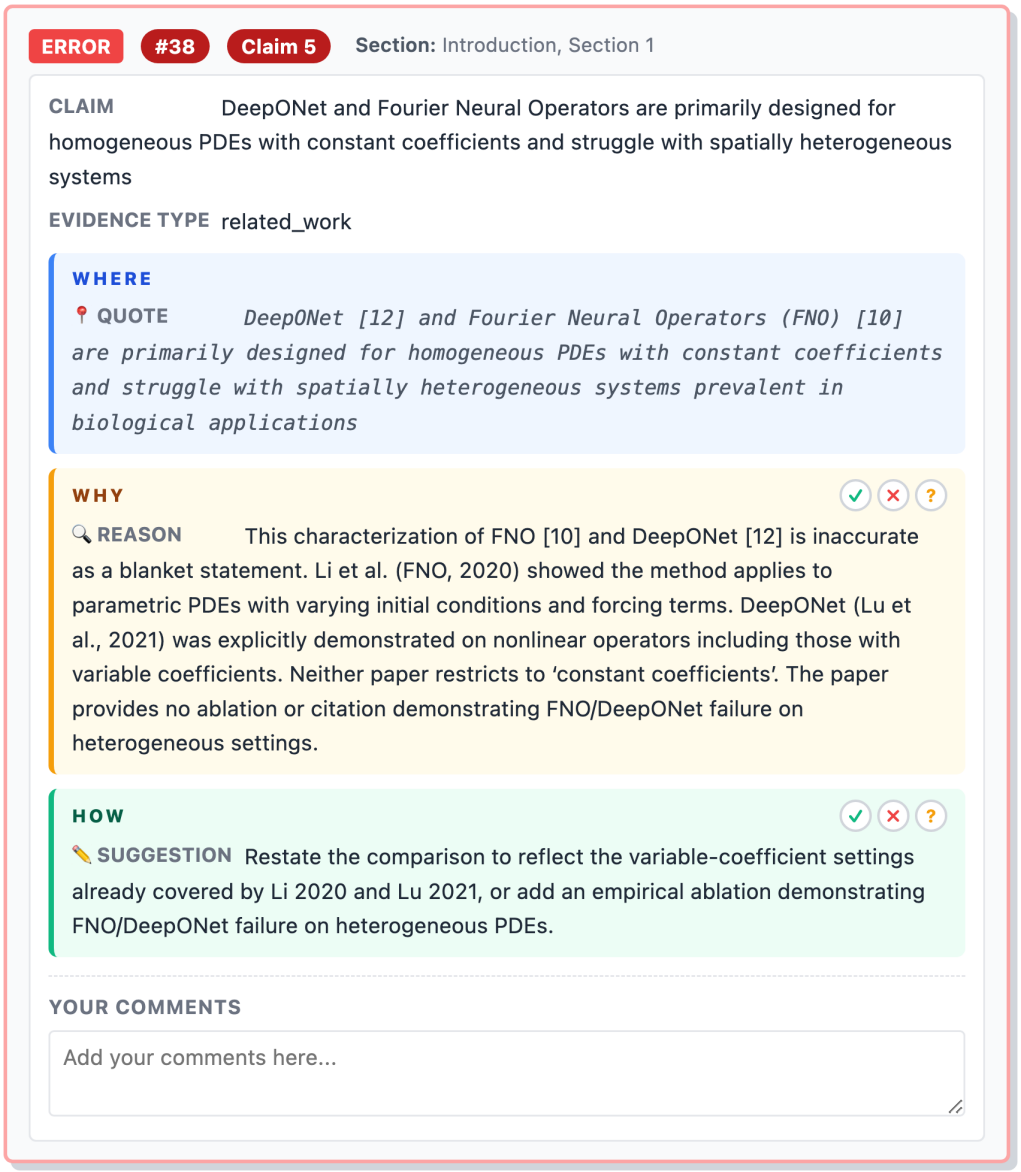
LLM-Driven Discovery of High-Entropy Catalysts via Retrieval-Augmented Generation (a second Agents4Science accepted paper) states in its Methods section that materials are "encoded using *SciBERT* into 768-dimensional vectors". PaperDoctor's code verifier dispatches the embedding-model claim to the released repository and finds that all three relevant files, `code_data/scripts/embedding_indexing.py` (line 21), `code_data/scripts/rag_retrieval.py` (line 33), and `pipeline_config.yaml` (line 9), set `model_name='all-MiniLM-L6-v2'`. *SciBERT* is never imported anywhere in the codebase, and the produced vectors are 384-dimensional rather than 768. The discrepancy spans both the model identity and its dimensionality. This is a paradigmatic case of a mismatch that is invisible to reading-only review: the paper text reads cleanly, but the implementation differs substantively from what is described.

L2 Theory Verification: a bound stated without definitions.



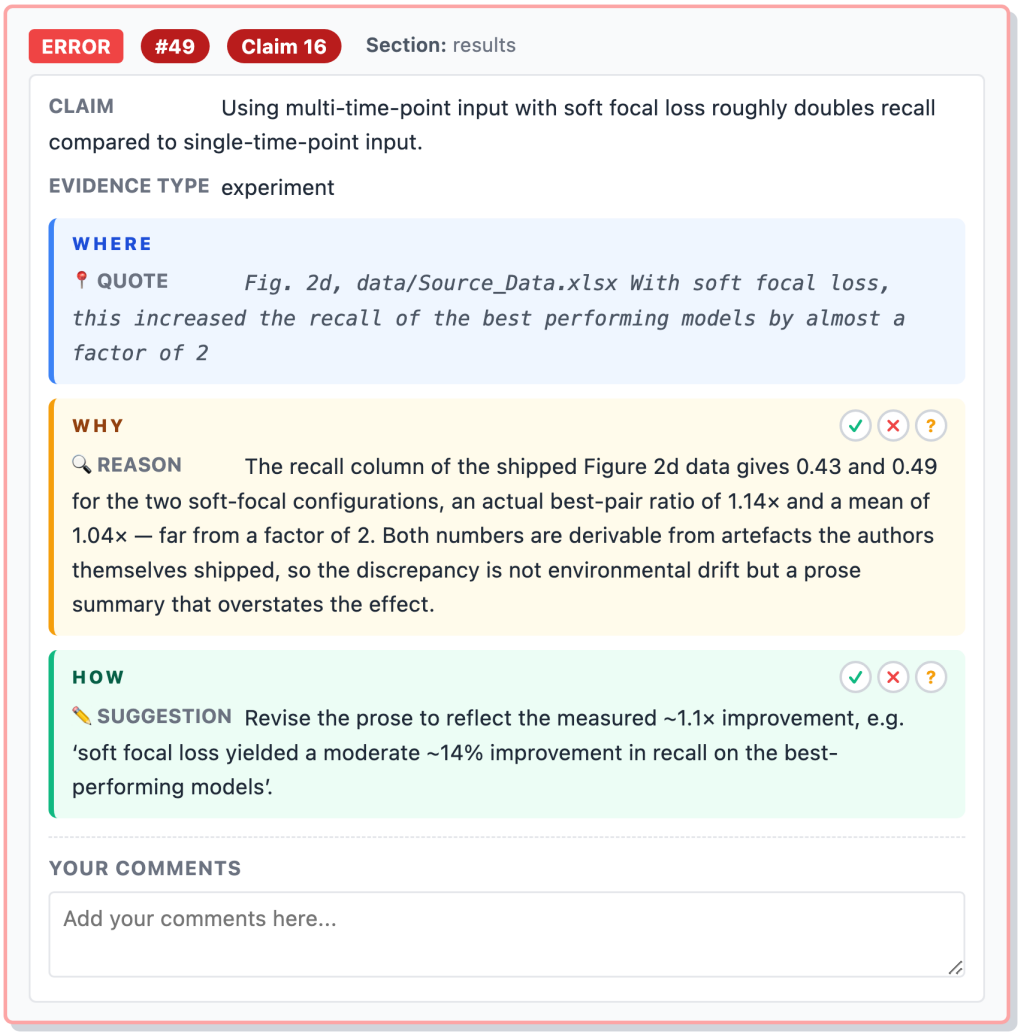
In *Neural Reaction-Diffusion Operators for Spatially Heterogeneous Tumor Modeling* (Agents4Science rejected), Section 4 states the approximation bound $\|G - G_\theta\|_{L^2} \leq C_1 W^{-\alpha/d} + C_2 L^{-\beta} + C_3 \|D - D_{\text{approx}}\|_\infty$ as if it were established. PaperDoctor's step by step re-derivation finds that none of the symbols composing the bound are introduced anywhere in the paper: W (presumably network width), L (presumably depth), the exponents α and β, and the constants C_1, C_2, C_3 all appear without definition, and no proof or pointer to an appendix is given. Without these, the bound cannot be checked for tightness or used to guide architecture design. On a first pass, such a gap looks like a finished theorem; only step by step re-derivation surfaces it.

L2 Literature Check: a novelty claim refuted by the cited works.



The same paper claims that "*DeepONet* [12] and *Fourier Neural Operators* (FNO) [10] are primarily designed for homogeneous PDEs with constant coefficients and struggle with spatially heterogeneous systems prevalent in biological applications." PaperDoctor dispatches this novelty assertion as a web search query, and the original method papers refute the framing directly: Li et al. (FNO, 2020, [10]) explicitly demonstrates the method on parametric PDEs with varying initial conditions and forcing terms, and Lu et al. (DeepONet, 2021, [12]) demonstrates nonlinear operators with variable coefficients. Neither prior work is restricted to the homogeneous setting the paper attributes to it. This pattern, where an inflated novelty claim is contradicted by the very references the paper cites, is exactly what a search-grounded literature check is built to catch.

L3 Reproduction: a "factor of 2" that is actually $1.14\times$.



In *Smart hybrid microscopy for cell-friendly detection of rare events*, a Nature Communications paper on mitochondrial imaging, the Results section claims that "soft focal loss … increased the recall of the best performing models by almost a factor of 2". The L3 stage rebuilds the analysis from the source data shipped with the paper (`data/Source_Data.xlsx`) and reads the recall column for Figure 2d directly. The two soft-focal configurations have contact-task recalls of 0.43 and 0.49, giving an actual best-pair ratio of $1.14\times$ and a mean ratio of $1.04\times$, far from the claimed factor of two. Both the claimed and reproduced numbers are derivable from artefacts the authors themselves shipped, so the discrepancy is not a matter of environmental drift but of how a quantitative claim was summarised in the text. The aggregate reproduction analysis (Sec. **??**) shows this kind of text-vs-data mismatch concentrated on Agents4Science papers; the case here shows that even peer-reviewed venues are not immune, and that reading the paper alone cannot easily catch it.

Takeaways. Across the five cases, two patterns recur. First, every finding is grounded in a concrete locus: a bibliography line, a config file, an equation, a cited paper's abstract, or a column of a shipped spreadsheet. The author can therefore audit each critique without re-reading the paper. Second, PaperDoctor catches the failure modes that reading alone misses: paper text that contradicts the released code or shipped data, undefined symbols hidden inside a clean-looking derivation, and novelty claims that a single web search refutes. Aggregate metrics show that PaperDoctor produces denser and better-grounded feedback than human or LLM reviewers; the cases here illustrate what PaperDoctor's feedback looks like in practice.

F Future works

Future directions stand out: (i) *Full paper-to-code reproduction.* Even when authors do not release a runnable codebase, an agent could synthesise a reference implementation directly from the paper itself, as PaperBench [30] does. One challenge is that this setting will yield even lower reproduction rates than current system. (ii) *Human-in-the-loop collaboration.* Our analysis (Fig. 6b) shows that PaperDoctor and human reviewers cover complementary aspects, motivating an interactive setup where authors accept or revise each suggestion and trigger affected skills to re-run, so that human judgement and agent coverage compound.