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

Kevin Qinghong Lin [1] **Siyuan Hu** [2] **Pan Lu** [3] **Yu Chen** [1] **Yanzhe Chen** [2] **Owen Queen** [3] **Yupeng Chen** [1] **Jialin Yu** [1] **Junchi Yu** [1] **Zifeng Ding** [4] **Yuanfeng Ji** [3] **Jindong Gu** [1] **Linjie Li** [5] **Mike Zheng Shou** [2] **Philip Torr** [1] **James Zou** [3]

Abstract

Autoresearch is accelerating the research pipeline, but they also let flawed claims enter the literature at scale. Human advisors catch such issues through careful, traceable feedback on in-progress drafts, yet this does not scale. We introduce PaperDoctor, an agent framework that shifts automated paper assessment from a judge to a diagnostician, with three key designs: **(i) Holistic hierarchical pipeline.** Papers are evaluated across writing, layout, references, code, theory, prior work, and experiments via three layers: L1 surface screening, L2 typed verifiers that route each claim to the skill owning its evidence, and L3 reproducers that rerun experiments by priority. **(ii) Evidence-grounded actionable feedback.** Each finding is a triple of observation (Why), location pointer (Where), and revision suggestion (How), making every critique auditable. **(iii) Effective experimental reproduction.** PaperDoctor selectively rebuilds and reruns experiments by claim importance and compute budget, surfacing reproducibility gaps invisible from the manuscript alone. On in-progress papers from 26 graduate students, PaperDoctor achieves 72% accuracy; on 40 human- and AI-authored manuscripts across ML, natural, and social sciences, it produces more grounded feedback than human and LLM reviewers, its findings track paper quality, and its reproduction stage surfaces limitations missed in the main text. We release a demo interface for authors to browse findings grounded in their paper. Our PaperDoctor is available at https://paperdoctor.github.io.

[1]University of Oxford [2]National University of Singapore [3]Stanford University [4]University of Cambridge [5]University of Washington. Correspondence to: Philip Torr <philip.torr@eng.ox.ac.uk>, James Zou <jamesz@stanford.edu>.

Accepted at the AI for Science Workshop at ICML 2026.

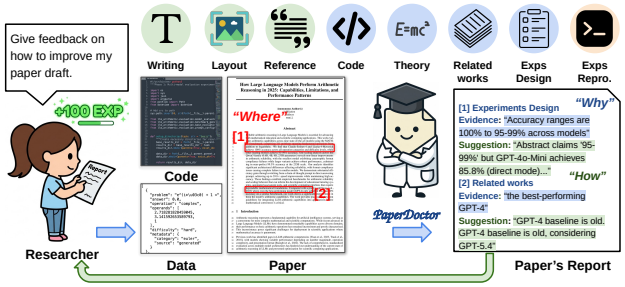


Figure 1. **PaperDoctor agent provides evidence-grounded, actionable feedback.** Given a paper along with its code and datasets, PaperDoctor returns a holistic report of findings across multiple dimensions (writing, citations, figures, equations, code, reproduction, and related work). Each finding pairs 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.

1. Introduction

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

Autoresearch agents (Lu et al., 2024; Yamada et al., 2025; Schmidgall et al., 2025; Tang et al., 2025; Bianchi et al., 2026) turn a proposed idea into a full manuscript, increasing the risk of producing work whose quality cannot be guaranteed, since such claims are often difficult 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 (D'Arcy et al., 2024; Jin et al., 2024; Gao et al., 2025; 2024; Zhu et al., 2025; Taechoyotin and Acuna, 2025) 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 junior researchers (Can and Walker, 2011; Steiss et al., 2024). They go through the draft carefully: circling a claim with no experimental validation, underlining a theorem whose 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 researchers 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 (Lu et al., 2024; Yamada et al., 2025; Schmidgall et al., 2025; Tang et al., 2025). Current agents either write the paper for the author (Tang et al., 2025; Bianchi et al., 2026) or grade it at the door (Liang et al., 2024; Rouzrokh et al., 2025; Jin et al., 2024; Tan et al., 2024; Li et al., 2026). Neither is what a researcher 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)** operates directly on the manuscript itself at low cost, primarily targeting objective issues in writing, formatting, references, and layouts. **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 executes 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 prioritizes 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 paper alone.

To validate the effectiveness of PaperDoctor, we first conduct a human study with junior researcher participants (26 graduated students), collecting their in-progress paper drafts and asking them to rate the feedback produced by PaperDoctor. PaperDoctor reaches 72% agreement with their ratings. Moreover, we evaluate PaperDoctor on 40 manuscripts covering machine learning, the natural sciences, and the social sciences, including both human- and AI-authored papers with accompanying code and data. This diverse benchmark allows us to systematically assess PaperDoctor's robustness across disciplines, 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's main body. In particular, by executing the accompanying code and rerunning 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. 9), 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.g.,* a brief description of the problem), 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. Notably, 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 (such as factual error), while a *warning* indicates uncertainty and flags the finding for further clarification, such as a human check. We enable LLM to determine them autonomously, in order to better leverage context.

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:

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

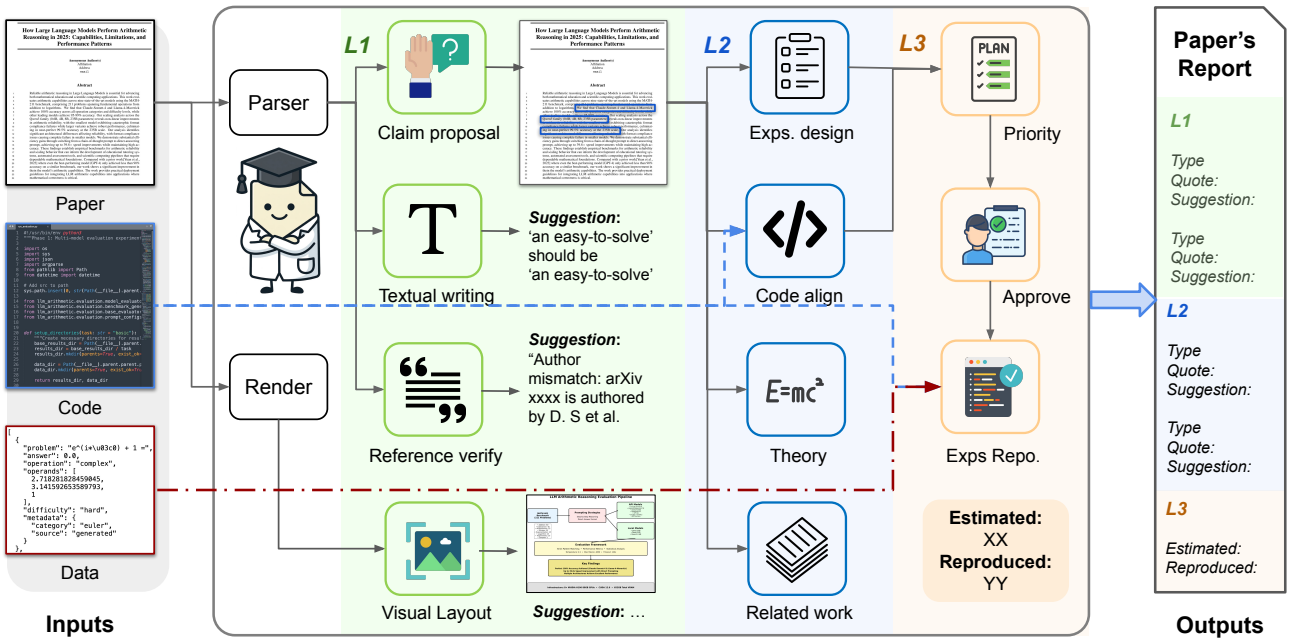


Figure 2. **Illustration of PaperDoctor pipeline.** PaperDoctor runs three diagnostic stages: L1 paper-only screening (Sec. 2.2), L2 typed claim verifiers (Sec. 2.3), and L3 prioritised experiment reproduction (Sec. 2.4), to produce evidence-grounded, actionable feedback on pre-submission papers.

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 2, 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 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 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.

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

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 $\mathcal{B}$, 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]`

Owing 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 $(\mathcal{P}_t, \{\mathcal{C}_j\})$, which are indexable during our parsing stage. 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 conflicts with the paper's 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. The 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 stage 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 their 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. PaperDoctor's Feedback Analysis

We examine PaperDoctor's feedback across three representative user groups (as illustrated in Figure 3): (a) authors, (b) reviewers, and (c) advisors. We design experiments to answer core research questions. ***Q1.*** How do different groups perceive PaperDoctor's feedback? ***Q2.*** How does PaperDoctor perform across papers of varying sources?

Settings. For authors, we conduct human studies with 26 graduate students, collecting their pre-submission paper drafts. For the PaperDoctor generated report, we let these authors score every finding and feedback as `{accepted, rejected, uncertain}`, separately. For reviewers, to study the performance across various paper types, we collect 40 papers from four diverse sources (as summa-

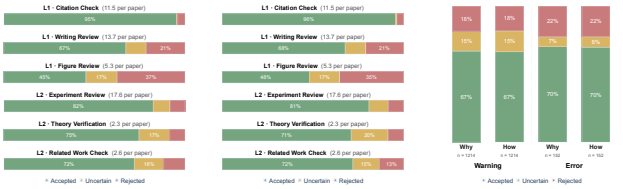
(a) **Junior students (authors)** *(b)* **Reviewers (evaluators)** *(c)* **Advisors (mentors)**

Figure 3. **We study how PaperDoctor's feedback interacts with three representative groups**: **(a)** junior students (authors), **(b)** reviewers (evaluators), and **(c)** advisors (mentors).

rized in Tab. 2). Ten come from Agents4Science (Bianchi et al., 2026), covering AI-generated scientific papers. Ten are human-written ICML papers with Oral and Spotlight awards. The remaining twenty are human-written papers from Nature Communications and Nature Human Behaviour (ten each), pulling the evaluation outside machine learning and into the natural and social sciences. For experts, which is the most challenging in terms of data collection, we prepare one in-progress paper paired with reviewer comments as well as senior advisors' suggestions.

3.1. Do *Authors* find PaperDoctor's feedback useful?

As shown in Fig. 4, we run PaperDoctor on 26 pre-submission papers and ask each author to label every finding on their own paper along two axes: `Why` — the *evidence* backing the finding (is the issue real and supported?) and `How` — the *feedback* offered with it (is the suggested fix the right one?). In total, authors review 1,642 findings and accept roughly 72% of them.



(a) `Why` (evidence): does PaperDoctor's diagnosis hold up against the author's own reading? Rows are L1 then L2, sorted by precision within tier; right-hand count is findings per paper.

(b) `How` (feedback): is the suggested fix something the author would act on? Same category list as left, with the same row order.

(c) Per severity, `Why` vs `How`. Four columns: `{Warning,Error}` × `{Why,How}`, comparing how the evidence and the feedback hold up inside each severity bucket.

Figure 4. Author votings (accepted / uncertain / rejected) on PaperDoctor's findings across 26 papers, viewed three ways: per category on `Why` *e.g.,* evidence (left), per category on `How` *e.g.,* feedback (middle), and per severity contrasting evidence vs feedback (right).

Where does the evidence hold up most? As shown in Fig. 4a, authors accept the majority of PaperDoctor's findings across every category, with `Why` acceptance ranging from 45% (figure) to 95% (citation). L2 dimensions, which target specific claim types such as theory and related work, achieve consistently high acceptance (72–82%), 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 finding volume (17.6 per paper), providing authors with a systematic reminder to revisit their experimental design.

Where does the evidence break down? *Figure dimension is the most rejected dimension*, with 37% of findings marked as Rejected. Three reasons stand out: (i) current vision models still struggle to parse complex scientific figures and tend to hallucinate; (ii) figure design is highly subjective, so automated judgments often diverge from what authors actually intend; and (iii) most authors spend significant effort drafting figures, so compared with text they are most sensitive to visual rather than textual feedback.

Does the feedback (`How`) follow the evidence (`Why`)? Comparing Fig. 4a with Fig. 4b, the two axes track each other almost exactly — per-category acceptance differs by at most a few percentage points on every category. Out of all findings rated on both axes, 96.1% have identical `Why` and `How` labels: when authors accept the evidence they almost always accept the feedback, and vice versa. The two axes therefore carry near-redundant signal in aggregate *e.g.,* once a finding is well-grounded, the suggested fix tends to be actionable too. However, the high consistency also serves as a reminder that *authors might not be neutral judges of criticism of their own work, and may tend to readily accept the proposed feedback.*

Does severity track human verdicts? Fig. 4c answers from the severity side. Within PaperDoctor's warning bucket (n=1,214), authors accept 67% on both `Why` and `How`; within the Error bucket (n=152), 70% on `Why` and 70% on `How`. Promoting a finding from 'Warning' to 'Error' therefore mainly pulls probability mass out of the Uncertain band rather than improving precision: the acceptance rate moves only $\sim$3 points but the rejection rate rises by 4, suggesting that *'Error' acts as a more polarizing signal*: more often correct, but met with stronger resistance when it is not, a calibration target for future iterations.

3.2. Do PaperDoctor's findings align with *Reviewers*?

Agreement between PaperDoctor and human reviewers. Beyond comparing what different types of reviewers focus on, we also measure the agreement between PaperDoctor's findings and human review comments across 40 papers spanning all domains. We decompose each review into atomic critique units using an LLM judge, then use a second LLM judge to determine semantic matches between PaperDoctor's units and human units. 'Claude-opus-4.7' is used for both the LLM judges. Figure 5 reports per-domain **pre-**

cision$= \frac{\text{\# matched agent's findings}}{\text{\# total agent's findings}}$, **recall**$= \frac{\text{\# matched human comments}}{\text{\# total human comments}}$, and their **harmonic F1**.

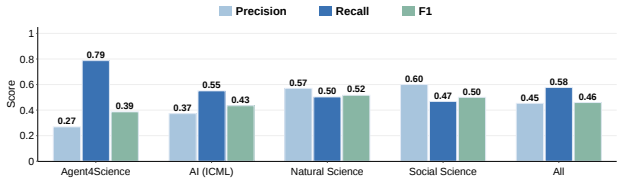


Figure 5. **Agreement between PaperDoctor and human reviewers**, measured as precision, recall, and F1 across four domains.

Overall, recall is higher than precision (0.58 vs. 0.45), confirming that PaperDoctor *covers many human-identified issues while also surfacing findings that reviewers do not mention*. The balance, however, shifts across domains: **(i)** on Agents4Science papers, whose reviews are produced by simple LLMs, recall reaches 0.79 but precision is only 0.27. This is because these AI-generated papers contain many surface-level issues that PaperDoctor flags but the LLM reviewers overlook. **(ii)** AI (ICML) papers show the same recall-heavy pattern, though with a smaller gap (0.55 recall vs. 0.37 precision). This suggests that PaperDoctor still produces richer feedback than conference reviewers. **(iii)** The pattern reverses for Nature and social science papers: precision rises to 0.57–0.60, while recall drops to approximately 0.47–0.50. This is likely because reviews of these papers contain high-standard, fine-grained critiques than those of Agents4Science and ICML papers.

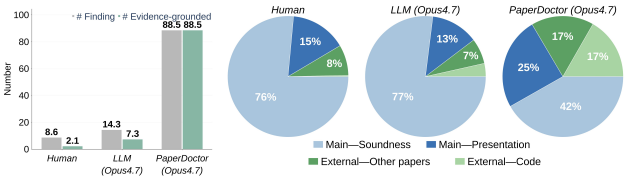


Figure 6. **Left: Finding number** (green means with evidence grounding), compared across human reviewers, an LLM reviewer, and PaperDoctor. **Right: Distribution of finding types.** *Main* refers to findings grounded in the paper body, *External* to those grounded in supplementary material, code, or other literature. *Soundness* covers method, theory, and experiments; *Presentation* covers writing and figures.

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. 'Claude-opus-4.7' is used for the LLM judge. We report results in Fig. 6. 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 88.5. The gap is by design: rather than taking each argument for granted, PaperDoctor surfaces dense 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 triggered the issue.

What do Humans, LLMs, and PaperDoctor focus on? In figure 6, we classified the review content into four buckets: soundness (method, experimental designs) and presentation (textual writing or visual presentation) within the paper body, and external checks (supplementary, other literature, or 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 67% of its findings fall on the paper body, while 17% check other literature works and 17% verify the released code. The sharpest gap is on code: human and LLM reviewers barely touch it, while 17% of PaperDoctor's findings verify the released code. 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.

3.3. Do PaperDoctor's findings align with *Advisors*? Case studies

(a) When each source engages

(b) What only the advisors brought

Figure 7. **Advisor feedback on one paper, against PaperDoctor and Human reviewers.** (a) The advisors comment continuously alongside the paper's 9 revisions; the automated PaperDoctor delivers one 238-finding pass; the reviewers land once (shaded band). (b) Example advisor comments (rephrased, de-identified) that no other source raised, or raised only weeks later.

The paper drew feedback from three structurally different

sources during one revision window: an automated per-block PaperDoctor (PaperDoctor, 238 findings in a single pass, 73 triaged by the author), two advisors (29 comments over five weeks of continuous email and chat), and three external reviewers (17 critique points across two workshops). Figure 7(a) shows they arrive in three distinct patterns: the advisors maintain a frequent revise-and-respond loop with the author, and on each new draft they follow up on the specific places that changed; the automated PaperDoctor delivers one instantaneous, high-volume batch; the reviewers come once, at the gate, and then fall silent, while the advisor stream continues past them.

What the advisors contribute is also structurally different (Figure 7(b)). Several of their concerns appear in no other source: that well-known test cases may already sit in the model's pretraining data, that the draft should be shared with professional practitioners for feedback before any wider release, and that an already-covered case study should be swapped for a fresher one. Where the human sources do overlap, the advisors get there first: the "system beats humans" over-claiming was flagged in the very first advisor reply, weeks before any formal review said the same, and the advisor's AI-judge concern directly caused the author to switch to a cross-family judge and rerun the evaluation. The PaperDoctor spends most of its volume on mechanical angles (claim grounding, experiment detail, surface writing) and the reviewers concentrate on rigor and polish, so the advisors are alone on training-data leakage and publishing strategy: volume, foresight, and gatekeeping only cover surface, substance, and strategy when combined.

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

Figure 8 breaks PaperDoctor's findings down by skill across four paper groups: human-written Nature Communications papers, human-written Nature Human Behaviour papers, human-written ICML papers, and AI-written papers from Agents4Science. Several observations stand out.

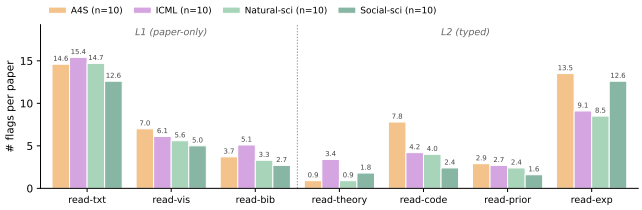


Figure 8. **Average findings per paper across paper categories.** We split papers into four groups by source: A4S (Agents4Science), ICML (Spotlight+Oral), Natural-sci (Nature Communications), and Social-sci (Nature Human Behaviour).

Surface writing quality is comparable across sources. All four sources show a similar density of writing issues (12.6–15.4 flags per paper), suggesting that surface-level writing is not the primary differentiator across paper sources. **Code verification separates AI-written papers from human-author papers.** The sharpest gap appears in code-related findings: A4S papers average 7.8 code flags per paper (51.0% flag rate), compared to 4.2 for ICML (29.3%). This reflects the common pattern of AI-written papers shipping code with configuration mismatches or incomplete implementations. Within L2, Experiment Design draws the most findings, 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. Overall, the gap between AI-written and human-written papers lies not in surface writing, where all four sources are comparable, but in code verification and experimental design.

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 produces 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 an 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 various disciplines, 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 reproducible research in the community.

Future works Two directions stand out: (i) *Full paper-to-code reproduction.* When authors release no runnable codebase, an agent could synthesise a reference implementation from the paper itself, as PaperBench (Starace et al., 2025) does, though this setting will yield even lower reproduction rates. (ii) *Human-in-the-loop collaboration.* Our analysis (Fig. 4) shows 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 human judgement and agent coverage compound. (iii) *Large-scale advisor-level studies.* Advisor feedback is scarce, rarely recorded systematically, and mostly informal. Collecting it at scale and grounding it against external reviewer and AI feedback would capture the insight that only experienced advisors provide.

References

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.

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.

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.

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.

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.

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.

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.

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.

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

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, 2023a.

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, 2023b.

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.

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

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.

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.

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

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

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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

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.

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.

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

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.

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

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.

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.

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.

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

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.

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.

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.

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.

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.

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.

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.

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*.

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.

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.

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.

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.

Appendix

A. Related Work

Autoresearch for Papers End-to-end autoresearch pipelines chain ideation, experimentation, and drafting into a single agentic loop (Lu et al., 2024; Yamada et al., 2025; Schmidgall et al., 2025; Tang et al., 2025; Bianchi et al., 2026), increasingly supported by language and coding agents (Hua et al., 2026; Yan et al.; Xiang et al., 2025) 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 (Liang et al., 2024; D'Arcy et al., 2024; Gao et al., 2024; Zhu et al., 2025; Taechoyotin and Acuna, 2025; Rouzrokh et al., 2025); multi-agent variants simulate the review cycle with role-specialised agents (Jin et al., 2024; Tan et al., 2024; Li et al., 2026); decomposition-based TreeReview (Chang et al., 2025) recursively asks sub-questions; and CycleResearcher (Weng et al., 2024) closes the loop via iterative preference optimization. Multimodal and multidisciplinary variants (Gao et al., 2025; Hong et al., 2025) read figures and tables; pre-submission assistance has been proposed as an ethically cleaner deployment (Foster, 2025). At scale, the Review Feedback Agent (Thakkar et al., 2025) was deployed on 20k ICLR-2025 reviews. Parallel audits document persistent failure modes: prompt-injection susceptibility, sycophancy, and poor novelty calibration (Li et al., 2025; Zhuang et al., 2025). 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 (Xu et al., 2026; Naser, 2026); dedicated verifiers (Yuan et al., 2026; Wu et al., 2025) and attributed-generation frameworks (Gao et al., 2023b;a; Bohnet et al., 2022; Ravichander et al., 2025) supply the primitives for PaperDoctor's reference verifier. (ii) Theoretical-claim verification. LLM-based provers (Ren et al., 2025; Baba et al., 2025; Varambally et al., 2025; Ospanov et al., 2025) and agentic program verifiers (Tu et al., 2025) 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 (Chan et al., 2024; Huang et al., 2023; Tang et al., 2023) evaluate ML engineering on Kaggle- and repo-scale tasks, while (Jimenez et al., 2023; Siegel et al., 2024; Chen et al., 2024; Wijk et al., 2024) extend this to software, computational, scientific, and frontier-R&D settings, with top agents still below 40% execution accuracy (Hua et al., 2026; Yan et al.; Xiang et al., 2025). 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. Experimental Reproduction Analysis

B.1. How is an L2 claim transformed into actual execution?

Figure 10 shows L2 experimental claims forwarded to L3. PaperDoctor surfaces 43.8 experimental claims per paper. 33% are routed to L3 with a reproduction plan, while the rest are flagged as not requiring re-execution, *i.e.,* simple inference checks or claims already self-verified within the paper. Of the routed plans, only 2.4 pass the feasibility check and are ready to run, while 12.1 are blocked, most often by missing model weights, data access, or heavy compute requirements. After human approval, 5.4 are eventually attempted, leading to 3.0 passes and 2.1 warnings per paper. End-to-end, 11.5% of L2 claims yield a reproduced number, while conditional on execution the success rate (pass plus warning) is 93.5%.

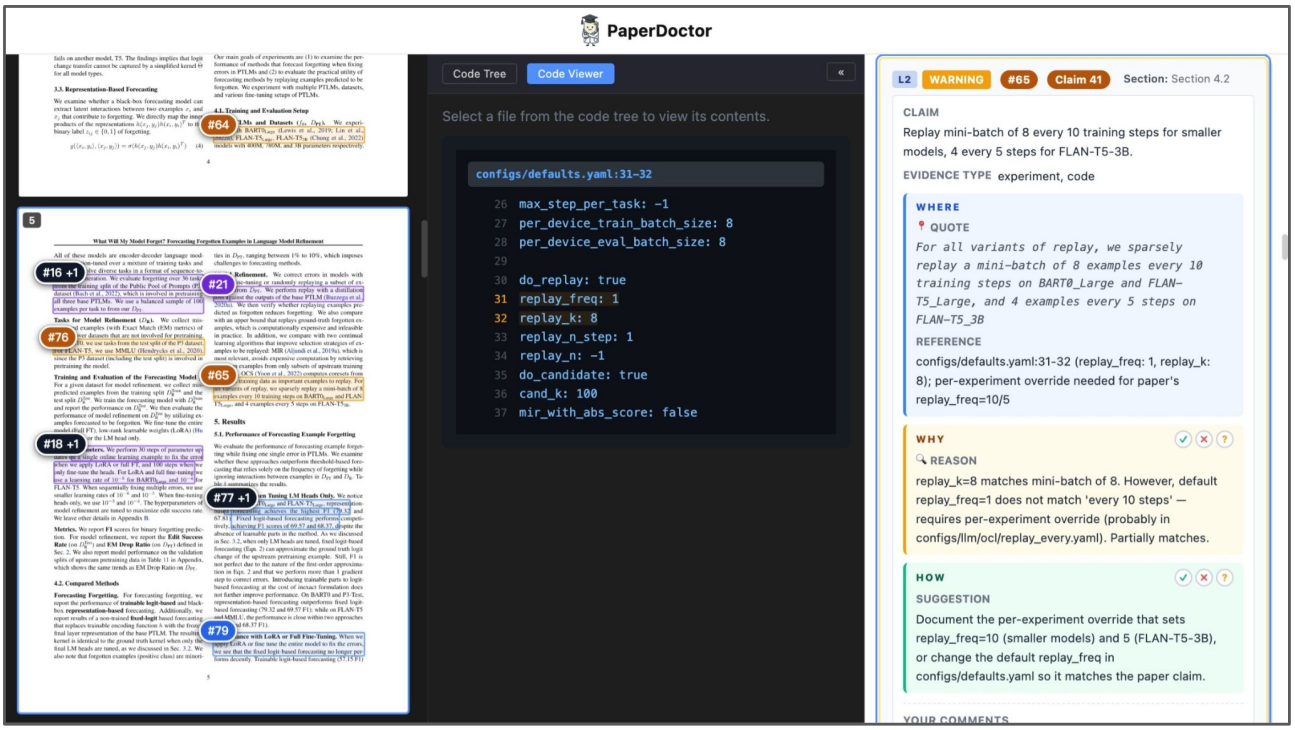


Figure 9. **The PaperDoctor Diagnosis Interface.** Authors upload their paper and code, and PaperDoctor returns the holistic report. **Left:** the paper, with each finding overlaid as a numbered, color-coded bubble anchored to the exact span it critiques (Where). **Middle:** the code viewer, so findings 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.

Table 1. **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-progress papers**: can the agent support or focus on these in-progress manuscripts. ✓=full, ✓=partial, ✗=absent.

System	Feedback Form			Assessment Coverage				Focus
	Reviewer Report	Grounded Evidence	Revision Suggestion	Text (LLM)	Visual (VLM)	Code Imple.	Exp. Repro.	In-progress Papers
Autoresearch								
AI Scientist (Yamada et al., 2025)	✓	✗	✗	✓	✗	✓	✗	✗
Peer Review								
MARG (D'Arcy et al., 2024)	✓	✗	✓	✓	✗	✗	✗	✗
AgentReview (Jin et al., 2024)	✓	✗	✓	✓	✗	✗	✗	✗
Reviewer2 (Gao et al., 2024)	✓	✗	✓	✓	✗	✗	✗	✗
DeepReview (Zhu et al., 2025)	✓	✗	✓	✓	✗	✗	✗	✗
TreeReview (Chang et al., 2025)	✓	✓	✓	✓	✗	✗	✗	✗
CycleResearcher (Weng et al., 2024)	✓	✗	✗	✓	✗	✗	✗	✗
MMReview (Gao et al., 2025)	✓	✓	✗	✓	✓	✗	✗	✗
Verification								
CiteAudit (Yuan et al., 2026)	✗	✓	✗	✓	✗	✗	✗	✗
PaperBench (Starace et al., 2025)	✗	✓	✗	✓	✗	✓	✗	✗
AutoReproduce (Zhao et al., 2025)	✓	✓	✗	✗	✗	✗	✓	✗
Feedback								
Review feedback (Thakkar et al., 2026)	✓	✓	✗	✓	✗	✗	✗	✗
Human advisor	✓	✓	✓	✓	✓	✓	✗	✓
PaperDoctor (ours)	✓	✓	✓	✓	✓	✓	✓	✓

Table 2. **Papers used in PaperDoctor evaluation. Ext. Reviews** indicates whether external reviews are available; **Advisors** indicates whether senior advisor comments are available.

Author	Source	Domains	Size	Status	Code	Ext. Reviews?	Advisors?
Feedback for Authors							
Humans	Graduated students	AI (Diverse fields)	26	In-progress	–	–	–
Feedback for Reviewers							
AI	Agents4Science	AI & Science	10	Oral/Spotlight/Poster	✓	✓	–
Humans	ICML	AI (ICML)	10	Oral/Spotlight	✓	✓	–
Humans	Nature Communications	Natural Science	10	Public	✓	✓	–
Humans	Nature Human Behaviour	Social Science	10	Public	✓	✓	–
Feedback for Experts							
Humans	PhD.	Social Science	1	In-progress	✓	✓	✓

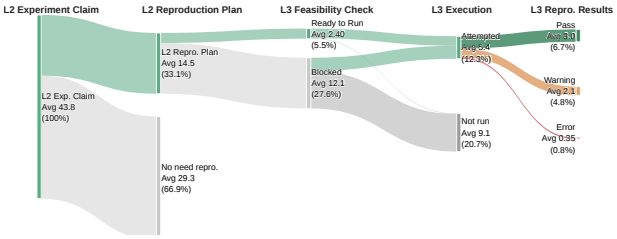


Figure 10. **End-to-end reproduction funnel across 40 papers.** PaperDoctor surfaces 43.8 experimental claims per paper on average; 14.5 receive a reproduction plan, 2.4 clear the feasibility check, and 5.4 are ultimately attempted. Of those that run, 2.95 pass and 2.1 yield warnings. The end-to-end yield is 11.5% reproduced claims per paper, or 93.5% conditional on execution.

B.2. Correlation between L2 priority and L3 reproduction success rate?

Figure 12a breaks down L3 outcomes by L2-assigned priority. Pass rates are highest for High-priority claims at 25.6%, though they do not fall monotonically: Low (21.7%) sits above Medium (13.8%). 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. Medium-priority claims, by contrast, are typically auxiliary experiments such as ablation sweeps or sensitivity analyses, for which authors state a target but seldom release a runnable script; they are blocked most often of all (40%, against 21% for High).

B.3. Correlation between reproduction mode and success rate?

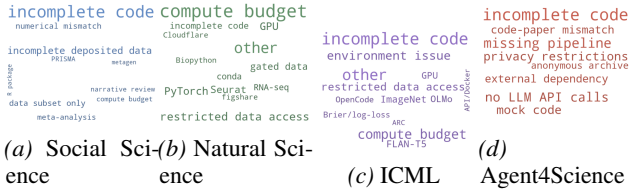
Figure 12b breaks the same outcomes down by reproduction mode. Pass rates differ several-fold: 28.7% for Eval (running released checkpoints), 19.6% for Other (largely the social-science analyses), and only 7.0% for Train. **Training is by far the hardest mode**, with 72% of plans blocked or never reached before any code runs, because training simultaneously requires the right environment, the data, and serious compute.

B.4. How does reproduction differ across paper sources?

Figure 12c breaks reproduction down across four paper sources. Plan density varies widely, from 18.1 and 16.7 plans per paper for Social-sci and ICML down to 8.6 for A4S, reflecting how much of each domain's claims map to concrete numerical targets: **social-science papers are dominated by statistical tests**, while **AI-written papers contain very few experiments** overall and are often short-paper format. In Fig. 12d, **Conditional pass rates (defined as Pass / Attempted)** tell a different story: A4S leads at 72.2%, followed by Social-sci at 61.8% and Natural-sci at 55.4%, while ICML papers fall to 35.8%. Two failure modes are visible. A4S struggles before execution, where many plans are blocked or never reached, but the few that do run almost always match. ICML struggles after execution: a fair share of its plans run, but the numbers often diverge from what the paper reports, as reduced-scale or single-seed training drifts from the published figures. Notably, **"Pass / Attempted"** measures how often a paper's experiments hold up *once execution is feasible*. Across all four sources the pass-or-warning rate stays above 90%, suggesting that *the bottleneck for reproduction is reaching execution, not the execution itself.*

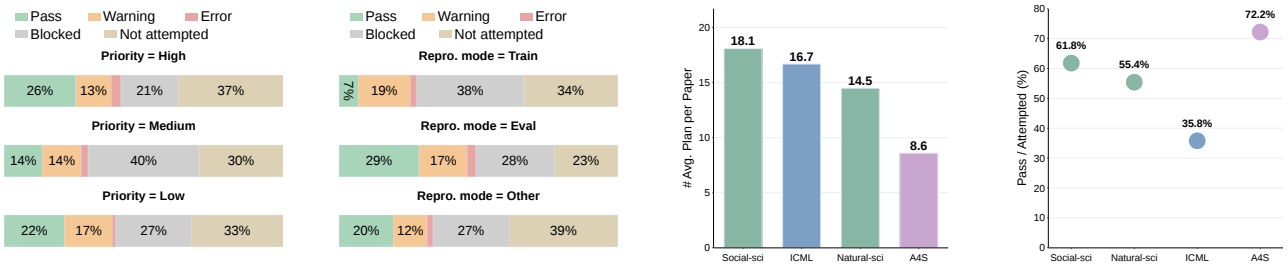
B.5. What kinds of failures dominate each domain?

The failure profile changes sharply by domain. Social Science failures are overwhelmingly artifact-completeness problems: missing scripts account for 84.1% of blocked or errored plans, with incomplete deposited data adding another 11.6%. Natural Science fails for a different reason: compute-heavy bioinformatics pipelines dominate at 60.9%, with restricted data access still visible at 13.0%. ICML is less concentrated. Missing code remains the largest category at 42.9%, but a long tail of paper-specific blockers, compute cost, and environment issues accounts for much of the rest. Agents4Science shows a smaller but consistent pattern of omitted configuration, pipeline, or data files. Overall, incomplete code remains the largest cross-domain bottleneck at 56.3%, but the route to failure is domain-specific rather than uniform.



(a) Social Science *(b)* Natural Science *(c)* ICML *(d)* Agent4Science

Figure 11. **Distribution of reasons why PaperDoctor fails at the reproduction stage.**



(a) By L2-assigned priority. *(b)* By reproduction mode. *(c)* Avg. number of L3 plans. *(d)* Conditional pass rate$=\frac{\#\text{Pass}}{\#\text{Attempted}}$.

Figure 12. **L3 reproduction breakdowns (a)** Pass rate by L2 priority. **(b)** Pass rate by reproduction mode. **(c)** Average L3 plans per paper, by paper source. **(d)** Conditional pass rate (pass/attempted), by paper source.