Analysis of Results in the ML Research Field

How well can an LLM decide the reproducibility of a paper?

Author: Andrei Opritoiu (A.A.Opritoiu-1@student.tudelft.nl) **Supervisors:** David M.J. Tax, Chenxu Hao, Hayley Hung, Nergis Tömen **Committee:** Klaus Hildebrandt

EEMCS Faculty
Delft, The Netherlands
June 21, 2026

1. INTRODUCTION

- **The Reproducibility Crisis:** A lot of submissions to conferences.
- **Growing Potential:** LLMs become better at reasoning.
- **Core Objective:** See if LLMs can determine the reproducibility of a paper.

RELATED WORK

- **Reproducibility Crisis:** ML research faces ongoing challenges with missing code, data, and hyperparameters [1], [2].
- **LLM Peer Review:** Early "ReviewerGPT" studies show LLMs can provide feedback but lack deep technical understanding [3].
- **NeurIPS 2024 Experiment:** Authors successfully "gamed" LLM checklist assistants using fabricated justifications that the model failed to verify [4].

REFERENCES

[1] M. Hutson, "Artificial intelligence faces reproducibility crisis," Science, vol. 359, no. 6377, 2018.

[2] H. Semmelrock et al., "Reproducibility in Machine Learning-Driven Research," arXiv:2307.10320, 2023.

[3] R. Liu and N. Shah, "ReviewerGPT? Exploratory Study on LLMs for Reviewing," arXiv:2306.00622, 2023.

[4] A. Goldberg et al., "LLM as Author Checklist Assistant: NeurIPS 2024 Experiment," PRC, 2024.

[5] J. Pineau et al., "Improving reproducibility in machine learning research (a report from the neurips 2019 reproducibility program)," JMLR, 22(164):1–20, 2021.

3. RESEARCH QUESTIONS

RQ1: Blind Trust & Self-Claims
Falsely labeling papers as "Reproducible" simply because the authors' self-claims say so, rather than verifying the evidence.

RQ2: Locating & Verifying Elements
Accuracy in locating and verifying code URLs, dataset citations, hyperparameter tables, and hardware descriptions within a paper.

RQ3: Vague Methodology & True Reproducibility
Distinguishing vague methodology descriptions to assess the true reproducibility of a paper.

4. DESIGN & PROMPTS

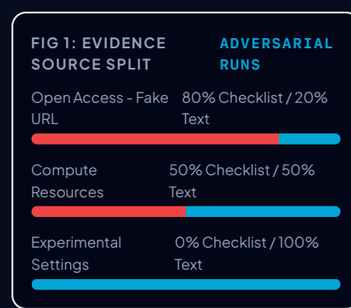
- **Checklist Trimming:** Trim author checklist.
- **Direct PDF Ingestion:** Feed the pdf directly into LLM.
- **Modular Architecture:** Evaluation split into three dedicated prompts: `LLM_PROMPT_OPEN_ACCESS`, `LLM_RESOURCE_COMPUTE`, and `LLM_SETTINGS`.
- **Expert Role-Prompting:** Domain-specific, expert NeurIPS peer-reviewer personas.
- **Negative Constraint:** Mandatory prompt directive: *"DO NOT USE THE AUTHORS ANSWER, YOU MUST CHECK FOR YOURSELF!"*.
- **Grounding Template:** Results in `exact_quote` extraction in JSON outputs to prevent semantic hallucinations.

5. RESULTS: Q1 & Q2

Q1: ADVERSARIAL AUDITS (N=30)

- **Design:** 30 test cases; 10 fake link contradictions, 10 compute omissions, 10 aligned controls.
- **Evaluation:** Compare against ground truth. Test if warning about author checklist improves accuracy.
- **80% Failure Rate:** Accepted fabricated code URLs when checking checklist assertions directly.
- **Warn Failure:** Text-only warnings failed to block model checklist navigation routes.
- **100% Core Cure:** Physical document trimming verified as only absolute safeguard.

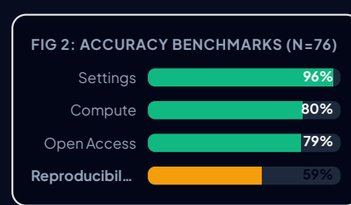
FIG 1: EVIDENCE SOURCE SPLIT — ADVERSARIAL RUNS



Q2: GRANULAR EXTRACTION (N=76)

- **96% Settings Accuracy:** Highly structured configurations extracted cleanly.
- **80% Compute Resources:** LLM started getting things wrong where previously in Q1 it did not, due to a larger prompt context window.
- **79% Open Access:** LLM cannot access links directly to verify their actual availability.

FIG 2: ACCURACY BENCHMARKS (N=76)



5. RESULTS: Q3 EMPIRICAL

Q3: EMPIRICAL BENCHMARK (N=40)

Methodology: Mapped the model's predictive reproducibility risk labels against real physical code execution outcomes archived in the ML Reproducibility Challenge (MLRC) database.

- **35% Match Rate:** Severely failed to predict actual physical human replication outcomes.
- **17.5% Environment Risk:** Blindness to deep hardware build dependencies.
- **Optimization Optimism:** Model mistook pristine writing structure for operational integrity.

FIG 3: MLRC PREDICTIONS: LLM VS HUMAN REVIEWERS (N=40)

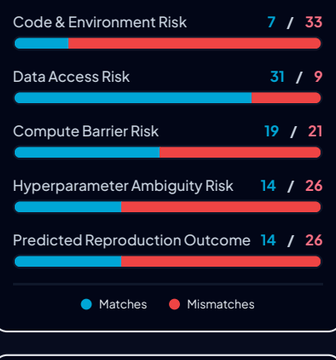


FIG 4: EMPIRICAL RISK VS OUTCOME MATRICES

Code & Env Risk (Human vs Predicted LLM)

	Low	Med	High
Low	5	1	2
Med	13	0	4
High	11	2	2

Reproduction Outcomes (Human vs Predicted LLM)

	Succ	Part	Fail
Succ	19	3	0
Part	21	2	0
Fail	1	1	0

X-Axis: Predicted (LLM) • Y-Axis: Expected (Human MLRC)

6. CONCLUSION

- Our findings show a clear gap between what an LLM can verify on paper and what actually happens when running the code.
- While the model is highly capable of scanning a manuscript to confirm that documentation exists, it is functionally blind to whether the associated code will execute successfully.
- Static document analysis remains fundamentally bounded by what is explicitly written, failing to account for silent infrastructure errors, missing dependencies, or socio-technical dynamics.
- To overcome this structural trap, automated peer review pipelines must transition from static text parsing to sandboxed dynamic execution layers.

7. DISCUSSION

- **Scale & Bottleneck:** Evaluated on focused manual sample sizes: N=30 adversarial, N=76 granular, and N=40 empirical.
- **Static Limits & Human Factors:** Static parsing misses socio-technical realities; human author collaboration swings replication success from 4% to 85% [5].
- **Structural Optimization Trap:** Systemic optimism leads to poor 17.5% environment risk detection; LLMs mistake clean tables and formatting layout for code validity.
- **Attention Dilution:** Monolithic prompts cause performance drops; search context degrades under multi-task queries, missing compute text over structured grids.

Source code at gitlab.tudelft.nl/cse3000/analysisMLresults