The paper has a GitHub, the GitHub has a README, the README has nothing: Reproducibility Signals for Review Support

Federico Bolelli*, ✉, Davide Santoli*, Kevin Marchesini, Luca Lumetti, and Costantino Grana

University of Modena and Reggio Emilia, Italy
{name.surname}@unimore.it

Abstract. Reproducibility policies promise "checkable" medical-imaging science, yet many submissions still ship unverifiable artifacts. Our analysis of 3 722 MICCAI papers shows code-linking rising from 51.8% (2021) to 72.5% (2025), but ∼13% of linked repositories are inaccessible or empty. We present PAPER-SNITCH, a reviewer-facing decision-support tool that turns these signals into an evidence-grounded report. PAPER-SNITCH parses PDFs, resolves and sanity-checks repositories, and applies policy-aware checklists aligned with MICCAI expectations, producing a review-time verifiability score decomposed into interpretable sub-scores plus criterion-linked excerpts and artifacts reviewers can inspect. It never executes untrusted code or attempts GPU-heavy reproduction, focusing instead on bounded, verifiable checks. We compare PAPER-SNITCH on 100 randomly sampled MICCAI 2025 papers with human annotators using shared evaluation criteria, indicating that automated, bounded checks can scale reproducibility screening while keeping final decisions with reviewers.

Keywords: Reproducibility Screening · Artifact Verification · Peer-review Support Tools · PAPER-SNITCH

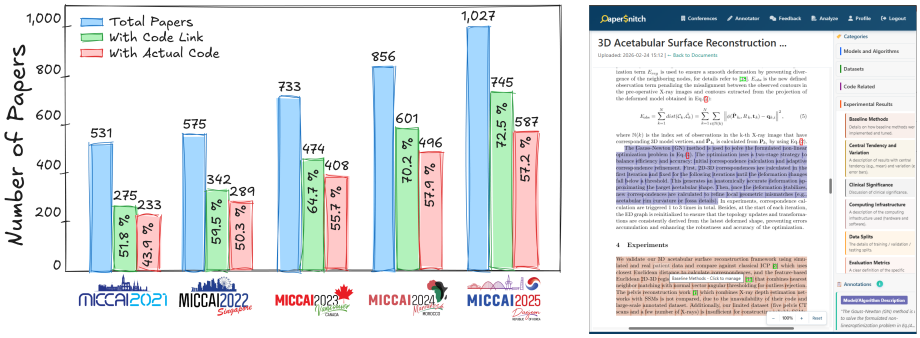
1 Introduction

Let us start by lowering your blood pressure: this paper does *not* introduce a new transformer variant, a clever loss, or a "lightweight" architecture that beats the state of the art (with a standard deviation, naturally, omitted for space constraints). Instead, we focus on a less glamorous but increasingly urgent problem for the MICCAI community: *how to keep our science checkable when the volume and complexity of scientific production are all accelerating at once.*

The current wave of (medical imaging) submissions is shaped by two trends: rapidly growing pipeline complexity and the widespread use of LLM-based code and text generation. While these tools can streamline engineering and documentation, they also amplify risks—overconfident narratives around fragile experiments, incomplete disclosure, and, in the worst case, low-quality manuscripts

*Equal contribution. Authors are allowed to list their names first on their CVs.

✉ Corresponding author: federico.bolelli@unimore.it



(a) Code availability in MICCAI proceedings (b) Findings

Fig. 1: (a) Analysis of 3 722 papers showing total publications (blue), papers with code links (green), and papers with verified accessible code (red). Code sharing has increased from 51.8% (2021) to 72.5% (2025), though ∼13% of linked repositories remain inaccessible or empty. Overall, 65.5% provide links, only 54.1% have actual code (see Sec. 3.2). (b) Evidence-highlighted paper view with criterion-linked excerpts.

that strain peer review [6,19]. Reviewers are asked to judge technical novelty, clinical relevance, evaluation rigor, and ethical considerations under strict time constraints, while reproducibility requirements add another axis of responsibility.

Reproducibility policies exist precisely to make this feasible. MICCAI provides explicit guidance [18] on what should be made available to support reproducible research (e.g., code, trained models where appropriate, documentation of data and evaluation, and clear statements describing what is and is not released). Other computer vision and pattern-recognition venues increasingly complement such guidance with *badges* or artifact evaluation tracks [1,10] that recognize verifiable releases and incentivize authors to produce them. The intent is clear: shift the norm from "trust me" to "here is what you need to check."

Yet, in practice, the weakest link is not the policy but the workflow (Fig. 1a): many papers still provide no code, or promise a release "after acceptance" that never materializes in the camera-ready [2,8]. Others provide a repository link that is empty, private, missing critical components, or contains little more than a placeholder README [3,13,24]. Even when artifacts are provided, a reviewer cannot realistically execute arbitrary code, resolve dependencies, acquire datasets, or reproduce GPU-heavy training. Consequently, compliance becomes difficult to assess, and the boundary between "not verifiable within review constraints" and "not sufficiently supported by provided artifacts" becomes blurred.

Crucially, a substantial part of this problem is *mechanical* and therefore amenable to automation: many review-time questions are objective verifications rather than scientific judgments. For example, does a linked repository exist and contain runnable code beyond a stub? Is there an explicit license? Are environment specifications provided (e.g., `requirements.txt`, `environment.yml`)? Are

pretrained weights and evaluation scripts available, and are paper claims traceable to concrete commands? Are data access instructions, splits, and evaluation protocols clearly specified, and do the artifacts match the venue's reproducibility statement? This motivates our central question: *can we leverage modern AI tooling to perform these bounded checks and present the results as decision support—not an automated gatekeeper—in a transparent, evidence-grounded report that highlights what is present, what is missing or unverifiable, and what still requires human interpretation?*

In this work, we introduce PAPER-SNITCH, a tool that supports review-time screening of reproducibility signals for medical imaging submissions by combining lightweight repository inspection, structured text analysis of the manuscript, and policy-aware checklists aligned with MICCAI's stated expectations [18]. The tool outputs *(a)* a review-time verifiability score (artifact completeness and consistency) decomposed into interpretable sub-scores, and *(b)* a set of *grounded* findings that can be inspected by the reviewer (Fig. 1b). The workflow is designed to be scalable and to integrate into a venue review pipeline as an auxiliary report, similar in spirit to artifact evaluation initiatives, but tailored to the time and safety constraints of double-blind peer review. It does not establish end-to-end reproducibility; it only checks what can be verified from provided artifacts under review-time constraints.

We start from MICCAI because its scope makes reproducibility both essential and challenging. Nonetheless, the approach is modular and can be adapted to other (computer vision) venues with different policies or requirements [1,10].

If this already convinced the reader, the tool implementation is available on GitHub,[1] while the live version is hosted at `https://paper-snitch.unimore.it`.[2]

2 Related Work

Reproducible Research, Checklists, and Artifacts. Reproducible computational science increasingly treats the contribution as a *research compendium*—paper, code, data (when shareable), and instructions to reproduce results [23]. Empirical studies report that many AI papers omit key details or artifacts needed to reproduce findings [5]. In response, major venues have introduced reproducibility checklists and artifact evaluation/badging to incentivize and signal verifiable releases [1,18,10,20,21]. However, verification remains costly and inconsistent, especially under review-time constraints.

Reproducibility in Medical Imaging. Biomedical image analysis faces additional barriers: restricted patient data, heterogeneous acquisition, and preprocessing choices that can dominate outcomes. Reporting standards and community guidance emphasize documenting data handling and evaluation protocols to improve interpretability and reproducibility [16]. Similar issues arise in MICCAI-style methodological work, where seemingly minor implementation differences can substantially change results while remaining underreported.

[1] `https://github.com/AImageLab-zip/paper-snitch`

[2] PAPER-SNITCH makes use of paid services; unauthenticated users can only visualize.

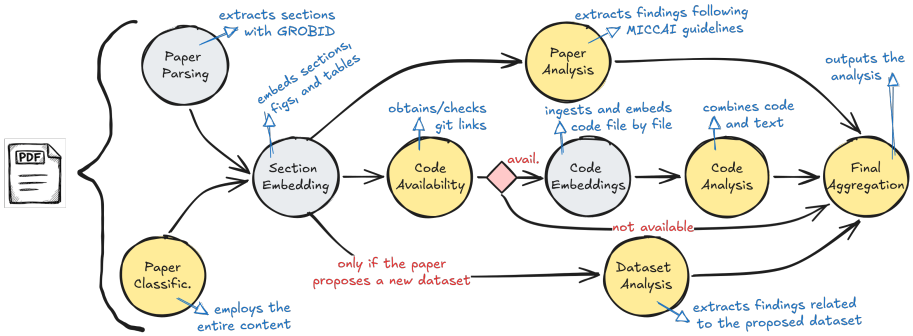


Fig. 2: End-to-end workflow of PAPER-SNITCH. Yellow nodes denote LLM calls; conditional branches are highlighted in red (diamonds and labels).

Automation for Review-time Screening. Recent discussions on generative AI in peer review highlight both opportunities for scalable consistency checks and risks around privacy, bias, and reliability [6,19]. Many reproducibility criteria are largely *verifiable* (artifact presence, environment specs, runnable evaluation, claim-to-command traceability) rather than requiring full re-execution. Our work targets this verifiable surface layer with policy-aware, evidence-grounded reports designed to support—not replace—human judgment.

3 Our Solution

3.1 Design Goals

PAPER-SNITCH is designed as *decision support* for review-time reproducibility screening (artifact verifiability), not an automated acceptance filter. This framing drives three non-negotiable constraints: *(i) evidence-grounded outputs*, every claim made by the tool must be traceable to an inspectable artifact (PDF text span, repository file, or link), *(ii) review-time safety*, the system never executes untrusted code and never attempts to reproduce GPU-heavy training runs. Instead, it performs *static* inspection, and *(iii) scalability and auditability*, the workflow must handle many papers concurrently, support partial completion, expose per-node failures, and preserve detailed intermediate artifacts so that chairs or a dedicated reproducibility track can audit how scores were produced.

3.2 System Overview

Reproducibility screenings are operationalized through the workflow in Fig. 2. The output is a compact report with decomposed scores and per-criterion evidence snippets that a reviewer can inspect. Nodes are detailed as follows.

Paper Parsing. Given a PDF, we extract a structured representation (TEI/XML) of the manuscript using GROBID [15], a machine learning library focused on

technical and scientific publications. The result is stored as section-wise text (and, when possible, figs./tabs. content and captions), so downstream steps can reference evidence at the level of concrete spans rather than free-form summaries.

Paper Classification. In parallel, we classify each paper as introducing a new *dataset*, proposing a new *method*, addressing *both* (dataset & method), or being purely *theoretical*, i.e., not requiring any code. This categorization is performed by a LLM and leverages the full content of the paper. This label has two explicit effects in the workflow: it determines which branches are applicable (e.g., pure theoretical papers can skip code analysis checks), and it selects the weighting used to compute the final score, which depends on the paper's category. What constitutes missing reproducibility information is contribution-dependent: the system should avoid penalizing inapplicable items.

Section Embedding. The manuscript is indexed for evidence retrieval [9]. All extracted sections are split into short, overlapping spans (2k characters with 20% overlap). Each span is embedded with a dense encoder and stored in a reusable vector index [12]. The rationale for this approach is twofold: criterion-level retrieval is cheaper and typically more stable than passing the entire paper to an LLM; chunking supports evidence localization and mitigates failures where information is missed or diluted in long inputs [14].

Paper Analysis. This branch implements the "paper-side" reproducibility checklist. Following [18], it evaluates 20 criteria grouped into three categories: *models*, *datasets*, and *experiments*. Each criterion has a precomputed embedding built from its name, detailed description, and examples. At runtime, for criterion embedding u and a section chunk embedding v, we compute cosine similarity $\mathrm{sim}(u, v) = \frac{u \cdot v}{\|u\|\|v\|}$ and retrieve the top-k most similar chunks, with a conservative similarity threshold and a token budget scheme. An LLM then evaluates *one criterion at a time* using only *(a)* the criterion definition and *(b)* the retrieved text spans, returning a structured record: *presence* $p \in \{0, 1\}$, *confidence* $c \in [0, 1]$, and quoted evidence/notes (i.e., what makes the LLM give that score with the corresponding confidence). Category scores are computed deterministically as $S_g = 100 \cdot \frac{1}{|I_g|} \sum_{i \in I_g} (p_i c_i)$, where p_i and c_i are respectively the present/absent decision and confidence for criterion i, and I_g is the set of criteria composing the category g. Finally, we combine category scores using paper-type-specific weights to obtain a paper-side score S_{paper}. Criterion-first evaluation yields interpretable failure modes ("missing split description" vs. "missing license"), enables per-criterion auditing, and keeps scoring stable: LLMs extract evidence; arithmetic is deterministic. The confidence is used to tune the score, assuming that lower values indicate a less clear presence.

Code Availability. This node gates the code branch. From the manuscript, we resolve repository links and perform lightweight sanity checks (reachable, non-empty, not a placeholder). If the code link is not available in the PDF, an online search might be performed depending on conference policies. Both operations, sanity check and online search, are supported by LLMs. If code is unavailable, the workflow marks the code branch as `skipped` and proceeds without it. Fig. 1a is generated from this node output.

Code Embeddings. When code is available, a searchable repository index is built in *two passes* to stay within a fixed token budget while preserving reproducibility-critical content. First, we ingest only the repository README(s) plus the full repository tree structure, including file sizes. We then ask an LLM to output an *ordered* list of include patterns (budgeted, e.g., ≤100k tokens total), prioritizing documentation, dependency specs (e.g., `requirements.txt`, `environment.yml`), and training/evaluation entry points, while excluding peripheral scripts (visualizations, extra baselines, benchmarks). Finally, we reingest *only* the selected files:[3] each file is chunked at word boundaries into non-overlapping ∼20k-character blocks and embeddings are computed. Chunks are stored with explicit positional metadata, i.e., *file_path*, *chunk_index*, *total_chunks*, to support evidence linking and retrieval-style lookups in downstream analysis. For integrity and cache invalidation, we compute and persist a per-file SHA-256 digest so reruns can detect code changes. Ingestion is guided by GitIngest [4]. We acknowledge that LLM-driven file selection may be fragile, and recommend venue-level guidelines that mandate clear, standardized repository entry points.

Code Analysis. During this step, the repository snapshot, the embedded code chunks, and paper spans produced by previous nodes are consumed. The analysis is decomposed into six components (a superset of MICCAI's code-oriented expectations [18]): *(i)* research methodology detection (e.g., *deep learning*, *machine learning*, *algorithm*, *simulation*, *data analysis*) together with applicability flags (e.g., *requires training*, *requires datasets*); *(ii)* repository structure and dependency specification quality; *(iii)* code components (e.g., training code, evaluation code, documented commands); *(iv)* released artifacts (e.g., checkpoints, dataset links, and coverage estimates); *(v)* dataset split specification and random seeds; *(vi)* documentation (e.g., README, reproduction commands).[4]

Each component is analyzed by a separate, targeted LLM call with structured output. Critically, the LLM input is jointly grounded in *repo evidence* and *paper claims*: for each component we assemble *(a)* the repository summary and file tree, *(b)* the most relevant embedded code/documentation chunks (taken from *Code Embeddings*), and *(c)* the corresponding paper spans (taken from *Section Embedding*), so the model can leverage both data sources and flag mismatches between manuscript and repository rather than implicitly assuming consistency. Both *(b)* and *(c)* are retrieved by cosine similarity against a component-specific query, similarly to what is described in the *Paper Analysis* node.

Numeric scoring is fully deterministic. Given component outputs, we compute a raw breakdown over *(ii)—(vi)* scoring axes with *methodology-adaptive* maxima (e.g., max code-completeness is 3.0 if training is required, else 2.5; artifacts are weighted up to 2.5 when datasets/training are required, else 2.0; splits are weighted up to 2.0 only when splits are required). The overall score is then normalized as $S_{\text{code}} = 100 \cdot \sum_j b_j / \sum_j m_j$, where b_j is the achieved score on axis j

[3] The system is configured to always include critical filenames (e.g., eval.*, train.*).

[4] Components *(ii)–(vi)* are methodology-dependent: *(i)* determines applicability, expected evidence, and thus the relative importance of each component in scoring.

and m_j its maximum score under the detected methodology; per-axis normalized scores $100 \cdot b_j/m_j$ are also reported in output. Finally, recommendations are generated programmatically from missing flags (e.g., missing checkpoints when *requires training* is true), ensuring actionable feedback without relying on the LLM for arithmetic or policy logic.

Dataset Analysis. If the classification indicates a dataset contribution (e.g., the paper explicitly claims a new dataset), we run a dedicated dataset documentation checklist (10 criteria derived from [18]) using the same retrieval-based, structured-evaluation pattern as *Paper Analysis*, yielding S_{data}. Indeed, dataset papers have additional reproducibility obligations (collection protocol, annotation, access restrictions, ethical statements) that are orthogonal to "method" reproducibility and should be assessed explicitly.

Final Aggregation. Although intermediate scores are always reported, the final report merges criterion-level outputs from *Paper Analysis*, *Code Analysis* (if available), and *Dataset Analysis* (if applicable) into a single "evaluation details" artifact, stored for audit and human comparison. The overall score is a weighted average with paper-type-dependent weights (e.g., methodological papers might use $S_{final} = 0.6S_{paper}+0.4S_{code}$). Such weights should reflect a pragmatic review rationale and should be adapted to each conference.

During the aggregation stage, LLM is used *only* to generate a short qualitative synthesis (strengths/weaknesses) from already-computed results; it does not generate or adjust numeric scores. The final output includes overall and component scores, a concise narrative summary, a machine-readable JSON artifact with criterion-level decisions and evidence, and an interactive paper view with highlighted supporting spans (Fig. 1b).

3.3 Infrastructure

Operationally, PAPER-SNITCH is deployed as a containerized web service via Docker, so the full pipeline (web UI, workflow workers, parsers, and databases) can be brought up reproducibly with pinned images and isolated dependencies. An nginx reverse proxy serves static assets, forwarding API traffic to a Django application. Long-running analyses are executed by a pool of Celery workers. The analysis logic itself is expressed as a versioned graph in LangGraph (StateGraph) [11], compiled and executed inside Celery tasks.

4 Performance in the Field

We evaluate PAPER-SNITCH on N=100 randomly sampled papers from the MICCAI 2025 proceedings, stratified over the three main categories identified in the *Paper Classification* node (Sec. 3.2): *dataset*, *method*, and *both*.
For each paper, we run the full workflow twice, swapping only the LLM used in all generation steps: GPT-4o [7] and GPT-5 [22]. Embedding models, i.e., `text-embedding-3-small` with a fixed embedding size of 1 536, and deterministic scoring code are kept fixed across runs to isolate the effect of the LLM.

Table 1: Human-machine agreement and I/O k-token usage per analysis axis.

Analysis Category	GPT-4o				GPT-5			
	% Agr.	MCC	#IN	#OUT	% Agr.	MCC	#IN	#OUT
Code	90.70	0.81	95.37	1.33	84.80	0.68	126.75	2.65
Paper	77.54	0.57	37.91	2.81	66.28	0.38	37.91	5.78
Dataset	72.22	0.29	13.39	0.99	65.00	0.18	13.39	1.71
Global	80.40	0.60	78.80	3.51	70.89	0.43	78.80	7.98

Whenever configurable, structured-output calls use low temperatures (0.1–0.2) for near-deterministic decoding in criterion-level analyses, while synthesis tasks use slightly higher values (0.3) to improve readability. Additionally, given the MICCAI paper length, we set the retrieval depth to k=3 to ground the LLM decision to a concise shortlist of evidence spans.

Human Reference Scores. We compare automated scores with human judgments from two annotators with more than 3 years of experience. Annotators filled the same criterion/component rubrics exposed by the tool (20 paper criteria, 10 dataset criteria, and the code-component checklist) and evaluated separate paper batches, discussing ambiguous cases to reach consensus before finalizing the rubric scores. In Tab. 1, the agreement between automated and human labels is computed by treating each field as a binary decision (true/false). Agreement is computed independently for each of the three main analysis components (paper, dataset, and code) and also overall (pooling all fields across components): we report percent agreement (accuracy), defined as the fraction of fields where the automated label matches the human label, and Matthews Correlation Coefficient (MCC) [17]. In our randomly sampled pool of 100 papers, paper-type classification was always correct. Tab. 1 also reports average token usage. The cost per paper averages $0.23 for GPT-4o and $0.43 for GPT-5.

5 Discussion and Conclusion

This paper presents PAPER-SNITCH as a practical proof-of-concept for evidence-grounded reproducibility screening in medical imaging submissions. The system decomposes reproducibility into interpretable criteria, ties decisions to inspectable evidence, and produces auditable artifacts that can support human judgment. We emphasize that weighting choices, criteria, and thresholds should be tuned to each venue's policies and validated before high-stakes deployment.

Limitations. Our evaluation uses a simple binary scheme, which may not capture partial releases, restricted-access data, or underspecified settings; future work should evaluate richer schemes against expert judgments and reproducibility outcomes. The system also ingests untrusted text (e.g., README) and relies on LLM-guided indexing, making it susceptible to prompt-injection; mitigations already include schema-constrained outputs and deterministic cross-checks

against the repository tree. Finally, while LLM costs are non-trivial, they appear feasible at conference scale and can be reduced via batching/caching or on-premises/open-weight models when privacy or budget requires it.

Acknowledgments. This project has received funding from Fondazione di Modena, through the FAR 2024 (E93C24002080007), and from MUR, under the NRRP "Fit4MedRob-Fit for Medical Robotics" (PNC0000007).

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Association for Computing Machinery: Artifact Review and Badging. https://www.acm.org/publications/policies/artifact-review-and-badging-current (2020), accessed 2026-05-25
2. Chen, J., Qi, F., Chang, C., Hu, Q., Fu, K., Wang, X., Liu, K.: GLM-SFNet: Global-Local Vision-Mamba with Semantic Fusion for Medical Image Segmentation. In: Proceedings of Medical Image Computing and Computer Assisted Intervention – MICCAI 2025 (2025)
3. Dai, P., Ou, Y., Yang, Y., Jin, Z., Suzuki, K.: GoCa: Trustworthy Multi-Modal RAG with Explicit Thinking Distillation for Reliable Decision-Making in Med-LVLMs. In: Proceedings of Medical Image Computing and Computer Assisted Intervention – MICCAI 2025 (2025)
4. GitIngest: Turn any Git Repository into a Prompt-friendly Text Digest (2026), https://gitingest.com/, accessed 2026-05-25
5. Gundersen, O.E., Kjensmo, S.: State of the Art: Reproducibility in Artificial Intelligence. In: Proceedings of the AAAI Conference on Artificial Intelligence (2018)
6. Hoyt, R., Limon, A., Chang, A.: Generative AI and scientific manuscript peer review. Intelligence-Based Medicine **11** (2025)
7. Hurst, A., et al.: GPT-4o System Card. arXiv preprint arXiv:2410.21276 (2024)
8. Jung, S., Lee, D., Kim, W.H.: MindLink: Subject-agnostic Cross-Subject Brain Decoding Framework. In: Proceedings of Medical Image Computing and Computer Assisted Intervention – MICCAI 2025 (2025)
9. Karpukhin, V., Oguz, B., Min, S., Lewis, P., Wu, L., Edunov, S., Chen, D., Yih, W.t.: Dense Passage Retrieval for Open-domain Question Answering. In: Conference on Empirical Methods in Natural Language Processing (2020)
10. Kerautret, B., Bolelli, F., Colom, M., Lopresti, D. (eds.): Reproducible Research in Pattern Recognition, Lecture Notes in Computer Science, vol. 15705. Springer Cham (2025)
11. LangChain: LangGraph – Build Resilient Language Agents as Graphs (2026), https://github.com/langchain-ai/langgraph, accessed 2026-05-25
12. Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.t., Rocktäschel, T., et al.: Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. Advances in Neural Information Processing Systems (2020)
13. Liu, J., Li, H., Yang, C., Deutges, M., Sadafi, A., You, X., Breininger, K., Navab, N., Schüffler, P.J.: HASD: Hierarchical Adaption for Pathology Slide-Level Domain-Shift. In: Proceedings of Medical Image Computing and Computer Assisted Intervention – MICCAI 2025 (2025)

14. Liu, N.F., Lin, K., Hewitt, J., Paranjape, A., Bevilacqua, M., Petroni, F., Liang, P.: Lost in the Middle: How Language Models Use Long Contexts. Transactions of the Association for Computational Linguistics (2024)
15. Lopez, P.: GROBID: Combining Automatic Bibliographic Data Recognition and Term Extraction For Scholarship Publications. In: International Conference on Theory and Practice of Digital Libraries (2009)
16. Maier-Hein, L., Reinke, A., Kozubek, M., Martel, A.L., Arbel, T., Eisenmann, M., Hanbury, A., Jannin, P., Müller, H., Onogur, S., et al.: BIAS: Transparent reporting of biomedical image analysis challenges. Medical Image Analysis **66** (2020)
17. Matthews, B.W.: Comparison of the predicted and observed secondary structure of T4 phage lysozyme. Biochimica et Biophysica Acta (BBA)-Protein Structure **405**(2) (1975)
18. MICCAI: Paper Submission Guidelines for Reproducible Research (2025), `https://conferences.miccai.org/2025/en/PAPER-SUBMISSION-GUIDELINES.html#reproducibleresearch`, accessed 2026-02-06
19. Naddaf, M.: AI is transforming peer review—and many scientists are worried. Nature **639**(8056) (2025)
20. NeurIPS: Paper Checklist Guidelines (2025), `https://neurips.cc/public/guides/PaperChecklist`, accessed 2026-05-25
21. Pineau, J., Vincent-Lamarre, P., Sinha, K., Larivière, V., Beygelzimer, A., d'Alché Buc, F., Fox, E., Larochelle, H.: Improving Reproducibility in Machine Learning Research (A Report from the NeurIPS 2019 Reproducibility Program). Journal of Machine Learning Research **22**(164) (2021)
22. Singh, A., et al.: OpenAI GPT-5 System Card. arXiv preprint arXiv:2601.03267 (2026)
23. Stodden, V.C.: Reproducible Research: Addressing the Need for Data and Code Sharing in Computational Science. Computing in Science & Engineering **12**(5) (2010)
24. Zhu, Z., Wang, J., Jiang, Y., Han, T., Huang, Y., Zhang, A., Yang, K., Luo, M., Liu, Z., Duan, Y., Ni, D., Tang, T., Yang, X.: Hierarchical Corpus-View-Category Refinement for Carotid Plaque Risk Grading in Ultrasound. In: Proceedings of Medical Image Computing and Computer Assisted Intervention – MICCAI 2025 (2025)