Agent-Based Auditing of Mathematical Proofs in Research Papers

Hieu N. Nguyen [1] **Rui Zhang** [1]

Abstract

We present an agent-based framework for auditing mathematical proofs directly from informal scientific papers, eliminating the need for strict formalization required by traditional theorem provers. The framework constructs a dependency graph across definitions, propositions, lemmas, and theorems to enable local verification—such as identifying logical gaps and hidden assumptions—alongside global consistency analysis to catch notation mismatches and cross-lemma dependency failures. To support iterative human-AI collaboration, the pipeline generates persistent, component-level audit reports. We evaluate our approach on a curated dataset of machine learning theory papers containing naturally occurring proof errors. Our framework demonstrates robust detection performance, suggesting that agentic structural auditing is a highly promising direction for improving the reliability of both human-written and machine-generated mathematical research.

1. Introduction

"Correcting an error may lead to interesting mathematics."
— Richard P. Brent

Auditing mathematical proofs in a research paper is an essential process for ensuring the rigor and clarity of theoretical computer science and mathematics. However, the manual effort required is so substantial that even peer-reviewed mathematical articles frequently contain errors (Brent, 2021; Wikipedia contributors, 2026). Even purely mathematical papers often conceal subtle ambiguities, logical leaps, unstated assumptions, and algebraic errors that evade casual scrutiny. Furthermore, the rate of published corrigenda remains significantly low (Grcar et al., 2013), highlighting a critical gap in our current validation pipelines.

[1]The Pennsylvania State University, State College, PA, USA. Correspondence to: Hieu N. Nguyen <hnn5071@psu.edu>.

Published at the ICML 2026 Workshop AI as a Tool for Mathematics, Computer Science, and Machine Learning, Seoul, South Korea. Copyright 2026 by the author(s).

Historically, the validation of mathematical literature has relied almost entirely on the decentralized, human-driven mechanism of peer review. More recently, formal verification systems such as Lean (de Moura et al., 2015), Coq (Team, 2024), Isabelle/HOL (Nipkow et al., 2002), and Mizar (Bancerek et al., 2015) have offered a way to verify with absolute logical certainty that a conclusion follows from its axioms. However, these systems require translating informal mathematical prose into a strictly formal language, which is a massive time investment. A general rule of thumb, the de Brujin factor, estimate that it typically takes about *10x* more effort to write a formal proof than it does to write the informal version for a paper (Wiedijk, 2000). Additionally, the learning curve for these tools is notoriously steep, and the errors raised during the formal proving process often fail to provide intuitive, actionable feedback to mathematicians.

The rapid advancement of Large Language Models (LLMs) and their emerging reasoning capabilities has spurred exploration into AI-assisted formal and informal auditing. However, even with substantial AI support, formalizing mathematical research often requires months or years of effort. On the other hand, informal auditing tools show promise in aiding authors during the drafting and refinement of complex theoretical work (Cohen-Addad & Woodruff, 2025). Yet, existing systems are typically either closed-source or limited to stateless, query-level interactions, lacking the persistent contextual understanding needed to track logical dependencies across an entire document.

To bridge this gap, we propose a stateful, interactive agentic framework designed to audit proofs comprehensively. Our main contributions are twofold. First, we introduce an agent-based auditing system, a stateful assistant that operates directly on informal mathematical documents within a native LaTeX project file structure to audit lemmas and theorems. Second, we provide an empirical analysis of practicality through a thorough evaluation of the system's performance on realistic theoretical research projects. This evaluation demonstrates the framework's effectiveness in detecting logical flaws and providing actionable feedback for researchers. We believe this approach represents a highly promising direction for improving the reliability of both machine-generated and human-written mathematical research.

2. Related Work

Formal theorem proving Formal theorem proving has a long history in automated reasoning and mathematical verification. Interactive proof assistants such as Coq proof assistant, Isabelle, and HOL Light provide rigorous machine-checked proofs through trusted logical kernels. Large-scale formal mathematics projects such as Mizar system and Metamath demonstrated that substantial mathematical corpora can be formalized and mechanically verified at scale (Matuszewski & Rudnicki, 2005; Hales, 2008). More recently, the emergence of Lean theorem prover has accelerated research in neural theorem proving and formal mathematics (De Moura et al., 2015). Despite their rigor, these systems fundamentally require fully formalized inputs. Translating informal mathematical writing into formal proof languages remains labor-intensive and constitutes a major bottleneck in formal verification workflows. Consequently, existing theorem provers are difficult to apply directly to LaTeX research papers written in natural mathematical language.

LLM-based formal proof generation. Benchmarks such as miniF2F (Zheng et al., 2022) and toolkits such as LeanDojo (Yang et al., 2023) have enabled large language models (LLMs) to interact with formal proof environments. Subsequent systems integrate LLMs with proof assistants for tactic generation, lemma retrieval, and iterative proof search (Polu & Sutskever, 2020; Azerbayev et al., 2024). Systems such as GPT-f (Polu & Sutskever, 2020), Llemma (Azerbayev et al., 2024), ReProver (Yang et al., 2023), and APOLLO (Ospanov et al., 2026) combine neural reasoning with formal proof environments to perform tactic prediction, premise retrieval, and proof repair. Recent advances have also introduced agentic theorem-proving frameworks such as ImProver (Ahuja et al., 2025) and the QED multi-agent proof system (An et al., 2026). However, these approaches' goal is to find a new proof for an existing problem, instead of auditing an existing proof, and often operate at the level of isolated theorem statements rather than full mathematical documents.

AI-Assisted Mathematical Review. Recent work has begun exploring the use of frontier language models as assistants for mathematical and theoretical computer science review workflows. Most notably, Google Research deployed a Gemini-based system for automated feedback generation during the STOC 2026 submission cycle (Cohen-Addad & Woodruff, 2025). The system provided pre-submission feedback on theoretical computer science papers, identifying issues ranging from inconsistent notation and calculation errors to subtle logical gaps in proofs (Cohen-Addad & Woodruff, 2025). However, the described framework primarily focuses on generating reviewer-style feedback and does not explicitly provide document-level proof structure, dependency graphs, or theorem obligation tracking. In contrast, our work emphasizes structured proof representations and agentic auditing mechanisms operating over interconnected lemmas, definitions, and assumptions.

3. The Mathematical Proof Auditing Pipeline

The proposed auditing framework operates in three phases: structural mapping, local component verification, and source-level validation. Each phase produces persistent intermediate artifacts inside an `audit/` workspace, enabling iterative verification without requiring the entire paper to remain inside a single model context window.

Phase 1: Structural Mapping. The auditor first indexes all mathematical items—such as definitions, assumptions, lemmas, and theorems—creating sequential audit files in the audit/ directory. Each file tracks the original statement, proof, prerequisites, and a placeholder for error analysis. Simultaneously, the framework initializes a centralized error registry (`error_table.md`). The system then generates a document-level dependency graph (see Figures 1 and 2), exported as both a Graphviz dot file and a prose summary. This phase concludes by scanning for global structural failures, such as circular dependencies or undefined notation, which are immediately logged in the central registry for subsequent validation.

Phase 2: Local component verification. Once the structural backbone is established, the framework performs detailed auditing by following the dependency order. For each component, the agent reconstructs the original proof as a sequence of *atomic logical steps*. This process minimizes implicit reasoning by decomposing sentences into a structured `Audit Proof` table consisting of `(Step ID, Claim, Premises, Justification)`. Suppose the original proof states: "By Lemma 2 and the definition of f, we have $f(x) \geq 0$ for all $x > 0$. Thus, by integrating both sides from a to b, we get $\int_a^b f(x)dx \geq 0$.", example of atomic decomposition is given in table 1.

ID	Claim	Premises	Justification
S1	$f(x) \geq 0$ for all $x > 0$	Lemma 2, def. f	Lemma 2 applies directly to f as defined
S2	$a > 0, b > a$	Thm. T1 Assumptions	Explicitly restate assumptions from theorem
S3	$\int_a^b f(x)dx \geq 0$	S1, S2, linearity of $\int$	Integrate both sides over $[a, b]$

Table 1. Atomic decomposition of the proof for Theorem T1.

The auditor also attempts to falsify the argument by checking for issues such as assumption drift, quantifier changes, dimension mismatches, and omitted boundary cases. Any step lacking rigorous justification is recorded

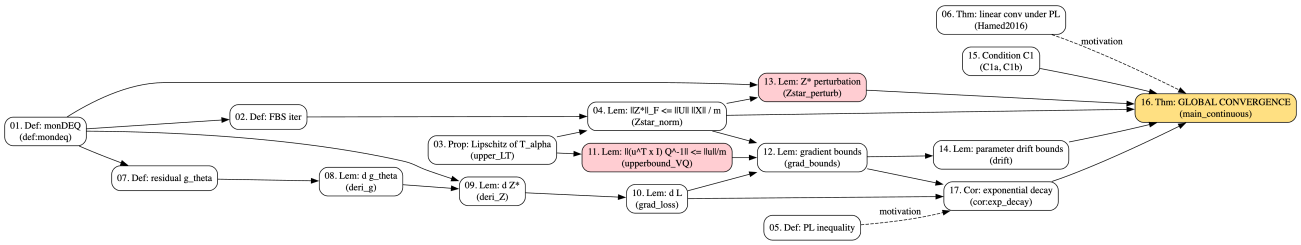

Figure 1. **Dependency graph of mathematical components in the proof auditing pipeline.** Nodes represent definitions, lemmas, and theorems, while directed edges encode logical dependencies between them. Highlighted nodes indicate the main theorem and detected error-prone lemmas identified by the auditor.

as a local warning and mirrored in the global error registry.

Phase 3: Error Validation. Once the local independent reviews are complete, the pipeline shifts to reconciliation, where the auditor cross-references the suspected errors from Phase 2 against the original LaTeX source code. This process begins with location and assessment, during which each flagged issue is traced to its exact line number in the source file. The auditor then seeks to confirm or refute the finding, determining whether the source contains a genuine logical flaw or a false positive. Finally, the framework updates all records within the Error Index and individual item files to reflect the verdict; confirmed errors are accompanied by explicit mathematical justifications.

Persistent Audit Artifacts. A key design principle of the framework is stateful persistence: instead of relying on a single transient interaction, the auditor incrementally writes results to a structured audit/ directory. This enables scalable processing of long papers beyond context limits and supports iterative multi-pass refinement. The persistent state also provides provenance tracking, reproducibility of detected issues, and human-readable access to intermediate reasoning. As a result, the audit workspace serves jointly as a proof reconstruction archive, error tracker, dependency graph, and structured review record, enabling stronger structural rigor than standard end-to-end LLM-based critique systems.

4. Auditing Example

This section details our pipeline's application to theoretical machine learning papers. Here, we analyze a manuscript proving a global convergence rate for the monotone deep equilibrium model (Winston & Kolter, 2020). Because this early draft contains known mathematical errors, it provides a realistic and challenging testbed for evaluating our framework while simultaneously mitigating concerns about memorization. We also present an analysis of a realistic theoretical machine learning research paper in Appendix Section C. Implementation details and reproducibility guidelines are provided in Appendix Section D[1].

4.1. Structure Graph

Figure 1 illustrates one of the primary outputs of Phase 1: a dependency graph of the document's mathematical items. In this graph, nodes represent specific elements (definitions, lemmas, and theorems), while edges denote logical dependencies. The proposed framework annotates the main theorem in yellow and flags lemmas containing critical errors in pink. We manually verified the structural correctness of the resulting graph.

4.2. Detected Errors

In this section, we highlight specific errors detected by the framework (indicated in red), accompanied by the context and error descriptions generated by the agent. Our qualitative analysis suggests that these descriptions offer actionable feedback for researchers to refine their proofs.

For quantitative evaluation, Table 2 presents the complete inventory of the 25 detected errors along with their manual verification status. Notably, 23 of the 25 detected errors were confirmed to be valid. In contrast, an LLM-as-a-Judge baseline using the same underlying model identified only 5 errors.

4.2.1. Lack of Rigor Error

The following definition define monotone deep equilibrium model. There is an abuse of notation in the definition which is highlighted. The auditor correctly identified and describe the error.

Definition 4.1 (Monotone DEQ)**.** The monDEQ considers a weight-tied, input-injected network with iterations of the form,

$$z^{[k+1]} = \sigma\left(W\mathbf{z}^{[k]} + U\mathbf{x}\right)$$

[1]The persistent artifacts are available at https://github.com/NNHieu/math_auditing_examples.

where $\mathbf{x} \in \mathbb{R}^d$ is the input, $U \in \mathbb{R}^{n\times d}$ is the input-injection weights, $W \in \mathbb{R}^{n\times n}$ is the hidden-unit weights, $\mathbf{z}^{(k)}$ is the hidden unit activation and $\sigma : \mathbb{R} \mapsto \mathbb{R}$ an elementwise non-linearity which is ReLU function in our case. The output of implicit layers is define as the fixed point $\mathbf{z}^*$ of the above update function

$$\mathbf{z}^* = \sigma\left(W\mathbf{z}^* + U\mathbf{x}\right)$$

To ensure the strong monotonicity condition, that $I - W \succeq mI$, the monDEQ parameterizes W as $W = (1-m)I - A^TA + B - B^T$, where $A, B \in \mathbb{R}^{n\times n}$. The output of the network is define as the linear transformation of the fixed point: $f(\theta, \mathbf{x}) = \mathbf{u}^\top \mathbf{z}^*$, where $\theta = (\mathbf{u}, A, B, U)$ are trainable parameters.

Description: "$I - W \succeq mI$" written as PSD, but W has skew part $B - B^\top$ so $I - W$ is non-symmetric; intended condition is operator strong-monotonicity (symmetrized).

4.2.2. Boundary Case Contraction Error

The following proposition presents an upper bound on the Lipschitz constant of an operator. The remark afterwards discusses the contraction of this operator given the value of α. This discussion includes a mistake, highlighted in yellow. The auditor correctly identifies and explains the error.

Proposition 4.2. *Let $I - W$ be m-strongly monotone with Lipschitz constant $L = \|I - W\|_2$. Then*

$$L[T_\alpha] \leq \sqrt{1 - 2\alpha m + \alpha^2 L^2}.$$

This implies that for $\alpha \in \left(0, \frac{2m}{L^2}\right)$,

$$L[T_\alpha] < 1.$$

Thus, the operator T_α is contractive for any $\alpha \in (0, \frac{2m}{L^2}]$, and this iteration is guaranteed to converge as long as the operator $I - W$ is Lipschitz and strongly monotone with parameter L.

Error Description: "Contractive for $\alpha \in (0, 2m/L^2]$" — at $\alpha = 2m/L^2$, $L[T_\alpha] \leq 1$ but not strictly < 1.

4.2.3. Critical errors

The following lemma is critical to the main result, as it is used to derive upper bounds on each gradient descent update and on the parameter trajectory length during training (see Lemma 11 in Figure 1). The correct inequality is $\|[I - W^\top J]^{-1} J\mathbf{u}\| \leq \frac{\|\mathbf{u}\|}{m}$, which holds under an active-subspace restriction. The lemma omits this condition, leading to an incorrect proof. The proposed framework correctly identifies this issue and marks it as a critical error.

For context, J is defined as $\mathrm{diag}(\sigma'(W\mathbf{z} + U\mathbf{x}))$.

Lemma 4.3. $\|[I - W^\top J]^{-1}\mathbf{u}\| \leq \frac{\|\mathbf{u}\|}{m}$

Proof. Let $\Pi = [J + \mu(I - J)]^{-1}$ with $\mu > 0$. Since $J_{ii} < 1 \forall i \in [n]$, we have $\Pi \in D^+$.

... Let $G = \left\{\left[I + \mu W^\top (I - J)\right]\Pi - I\right\}$ and $F(\mathbf{v}') = \left(I - W^\top\right)\mathbf{v}' - \mathbf{u}$.

The resolvent operator for G is $R_G = (I + \gamma G)^{-1}$, with $\gamma = 1$ then $R_G = \Pi^{-1}\left[I + \mu W^\top (I - J)\right]^{-1}$. The update of forward-backward splitting is

$$\begin{aligned}\bar{\mathbf{v}}^{k+1} =& R_G(\bar{\mathbf{v}}^k - \alpha F(\bar{\mathbf{v}}^k)) \\ =& \Pi^{-1}\left[I + \mu W^\top (I - J)\right]^{-1} \\ & \left\{\left[I - \alpha(I - W^\top)\right]\bar{\mathbf{v}}^k + \alpha\mathbf{u}\right\}\end{aligned}$$

From the proposition 4.2, we have $L[\left[I - \alpha(I - W^\top)\right]] \leq \sqrt{1 - 2\alpha m + \alpha^2 L[I - W^\top]^2}$. Take the limit $\mu \to 0^+$, we have $\|\Pi\| \leq 1$ and $\|R_G\| \leq 1$.

Same argument from lemma **??**, we have $\|\bar{\mathbf{v}}^*\| \leq \frac{\|\mathbf{u}\|}{m}$. Further,

$$\mathbf{v}^* = \Pi\bar{\mathbf{v}}^* = \left\{\left[I - \alpha(I - W^\top)\right]\bar{\mathbf{v}}^* + \alpha\mathbf{u}\right\}$$

when $\mu \to 0$). So $\|\mathbf{v}^*\| \leq \|\bar{\mathbf{v}}^*\| = \frac{\|\mathbf{u}\|}{m}$. □

E11-PI - Errors of omission: "Take the limit $\mu \to 0^+$, we have $\|\Pi\| \leq 1$." This is false. $\Pi = [J + \mu(I - J)]^{-1}$ has $\Pi_{ii} = 1/[J_{ii} + \mu(1 - J_{ii})]$. For ReLU's $J_{ii} = 0$, $\Pi_{ii} = 1/\mu \to \infty$ as $\mu \to 0^+$. The norm of Π diverges along inactive coords, so "$\|\Pi\| \leq 1$" in the limit is mathematically wrong.

E11-RG - Gap in logics: "$R_G = (I + \gamma G)^{-1}$, $\gamma = 1$, $R_G = \Pi^{-1}[I + \mu W^\top (I - J)]^{-1}$." No verification that G is maximally monotone, hence no firm-non-expansiveness of R_G.

5. Conclusion

We proposed an agent-based framework for auditing informal mathematical proofs in LaTeX documents. The system builds a dependency graph over definitions, lemmas, and theorems, enabling structured and iterative verification beyond stateless LLM critiques. By combining structural reconstruction, local verification, and global consistency checks, the framework identifies logical gaps, hidden assumptions, and notation errors, while maintaining traceable audit artifacts. Overall, this demonstrates that agentic auditing is a promising direction for improving the reliability of mathematical proofs in scientific research papers.

Impact Statement

This work aims to improve the reliability of mathematical and theoretical machine learning research by introducing an agent-based framework for auditing informal LaTeX proofs. Potential benefits include more reliable peer review, improved educational tools, and reduced effort in manual proof checking. However, limitations include possible false positives or negatives and the risk of over-reliance on automated verification without sufficient human oversight.

References

Ahuja, R., Avigad, J., Tetali, P., and Welleck, S. Improver: Agent-based automated proof optimization. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=dWsdJAXjQD.

An, C., Ye, Q., Pan, M., and Zhang, J. Qed: An open-source multi-agent system for generating mathematical proofs on open problems. *arXiv preprint arXiv:2604.24021*, 2026.

Anthropic. Claude code: An agentic coding tool that lives in your terminal. https://github.com/anthropics/claude-code, 2025. Accessed: 2026-06-25.

Azerbayev, Z., Schoelkopf, H., Paster, K., Santos, M. D., McAleer, S. M., Jiang, A. Q., Deng, J., Biderman, S., and Welleck, S. Llemma: An open language model for mathematics. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=4WnqRR915j.

Bancerek, G., Byliński, C., Grabowski, A., Korniłowicz, A., Matuszewski, R., Naumowicz, A., PaźK, K., and Urban, J. Mizar: State-of-the-art and beyond. In *Proceedings of the International Conference on Intelligent Computer Mathematics - Volume 9150*, pp. 261–279, Berlin, Heidelberg, 2015. Springer-Verlag. ISBN 9783319206141. doi: 10.1007/978-3-319-20615-8_17. URL https://doi.org/10.1007/978-3-319-20615-8_17.

Brent, R. P. Some instructive mathematical errors. *arXiv preprint arXiv:2106.07269*, 2021.

Cohen-Addad, V. and Woodruff, D. Gemini provides automated feedback for theoretical computer scientists at stoc 2026. Google Research Blog Post, 2025. URL https://research.google/blog/gemini-provides-automated-feedback-for-theoretical-computer-scientists-at-stoc-2026/.

De Moura, L., Kong, S., Avigad, J., Van Doorn, F., and von Raumer, J. The lean theorem prover (system description). In *International Conference on Automated Deduction*, pp. 378–388. Springer, 2015.

de Moura, L., Kong, S., Avigad, J., van Doorn, F., and von Raumer, J. The lean theorem prover (system description). In Felty, A. P. and Middeldorp, A. (eds.), *Automated Deduction - CADE-25*, pp. 378–388, Cham, 2015. Springer International Publishing. ISBN 978-3-319-21401-6.

Grcar, J. F. et al. Errors and corrections in mathematics literature. *Notices of the AMS*, 60(4):418–425, 2013.

Hales, T. C. Formal proof. *Notices of the AMS*, 55(11): 1370–1380, 2008.

Matuszewski, R. and Rudnicki, P. Mizar : the first 30 years. 2005. URL https://api.semanticscholar.org/CorpusID:1065140.

Nipkow, T., Wenzel, M., and Paulson, L. C. *Isabelle/HOL: a proof assistant for higher-order logic*. Springer-Verlag, Berlin, Heidelberg, 2002. ISBN 3540433767.

Ospanov, A., Farnia, F., and Yousefzadeh, R. APOLLO: Automated LLM and lean collaboration for advanced formal reasoning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026. URL https://openreview.net/forum?id=fxDCgOruk0.

Pham, H., Ta, T.-A., and Tran-Thanh, L. Pruning at initialisation through the lens of graphon limit: Convergence, expressivity, and generalisation. *arXiv preprint arXiv:2602.06675*, 2026.

Polu, S. and Sutskever, I. Generative language modeling for automated theorem proving. *ArXiv*, abs/2009.03393, 2020. URL https://api.semanticscholar.org/CorpusID:221535103.

Team, T. C. D. The coq proof assistant, September 2024. URL https://doi.org/10.5281/zenodo.14542673.

Wiedijk, F. The de bruijn factor, 2000.

Wikipedia contributors. List of incomplete proofs — Wikipedia, the free encyclopedia, 2026. URL https://en.wikipedia.org/wiki/List_of_incomplete_proofs. [Online; accessed 10-May-2026].

Winston, E. and Kolter, J. Z. Monotone operator equilibrium networks. *CoRR*, abs/2006.08591, 2020. URL https://arxiv.org/abs/2006.08591.

Yang, K., Swope, A., Gu, A., Chalamala, R., Song, P., Yu, S., Godil, S., Prenger, R., and Anandkumar, A. LeanDojo: Theorem proving with retrieval-augmented language models. In *Neural Information Processing Systems (NeurIPS)*, 2023.

Zheng, K., Han, J. M., and Polu, S. minif2f: a cross-system benchmark for formal olympiad-level mathematics. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=9ZPegFuFTFv.

A. Taxonomies of Errors in Informal Mathematical Proofs

While formal verification systems like Lean or Coq treat any deviation from logic as a kernel-level error, human-written LaTeX papers contain a spectrum of issues ranging from superficial notation slips to foundational logical collapses. Drawing on historical mathematical oversights and the error classifications identified by Brent (2021), we define the following categories that our pipeline is specifically designed to detect.

Typos are superficial errors in notation, including incorrect subscripts, variable swapping, or broken internal cross-references. While frequently seen in first-edition proofs and often considered "trivial," they can heavily obscure the path to verification.

Numerical or algebraic errors represent mechanical failures in computation, ranging from simple sign errors to more complex failures in matrix calculus. Historically, these encompass errors in implementing algorithms or relying on numerically unstable approximations.

Errors of omission occur when a proof fails to consider all possible cases, behaviors, or boundary conditions within a problem domain, ultimately leading to an incomplete proof. Historical examples include Henri Poincaré's initial omission of chaotic behavior in the three-body problem, and failures to account for integers with more than two distinct prime factors in early Prime Zeta function analyses.

Unwarranted or missing assumption errors arise when a mathematical condition, property, or constraint is assumed without explicit justification or proof. This often results in a proof that only holds for a specific subset of the claimed domain. For example, many flawed attempts at the Riemann Hypothesis rely on assuming properties of $\zeta(s)$ that hold for $\Re(s) > 1$ also apply to the critical strip $1/2 \leq \Re(s) \leq 1$.

Gaps in a proof are logical leaps where a claimed implication is not immediately obvious and requires intermediate lemmas that are missing from the text. A famous historical case is the gap discovered by Nick Katz in Andrew Wiles's initial 1993 proof of Fermat's Last Theorem, which required a year of additional work to bridge.

Misapplication of correct results happens when a valid theorem or lemma is utilized in a context where its underlying prerequisites are not satisfied, or where definitions differ subtly between the source and the current work. A historical instance is Paul Erdős applying a result outside its domain of applicability in a 1960 paper on integer multiplication.

Propagation of published errors occurs when an incorrect result is inherited from a previously peer-reviewed paper, creating an "error tree" where the current work's validity depends entirely on a flawed foundation. Historically, Léo Agélas fell into this trap by relying on an incorrect theorem by Vassilev-Missana to incorrectly claim a proof of the Generalized Riemann Hypothesis.

General lack of clarity or rigor plagues proofs that may be conceptually sound but are presented with ambiguous notation, missing steps, or hand-wavy transitions that make formal verification impossible. A notorious historical example is Mochizuki's claimed proof of the abc conjecture, which remains controversial due to its idiosyncratic and impenetrable notation.

B. Details Auditing Examples

Table 2. Comprehensive Audit of Mathematical Errors for the Example in Section 4. Of the 25 errors detected by the agent, 23 (92%) were manually verified as valid.

ID	Full Error Description	Status	Severity	Ver.	Type of Error
E01	"$I - W \succeq mI$" written as PSD, but W has skew part $B - B^\top$ so $I - W$ is non-symmetric; intended condition is operator strong-monotonicity (symmetrized).	▲Err	Minor	✓	Lack of rigor
E03	"Contractive for $\alpha \in (0, 2m/L^2]$" — at $\alpha = 2m/L^2$, $L[T_\alpha] \leq 1$ but not strictly < 1.	▲Err	Minor	✓	Typo
E04a	L'Hôpital step substitutes $\alpha = 0$ into the unsimplified derivative ratio; the cleaner version takes the limit of the simplified ratio. End answer $1/m$ is correct.	▲Err	Minor	–	Short cut notation

Continued on next page

ID	Full Error Description	Status	Severity	Ver.	Type of Error
E04b	"Differentiating and simplifying ... shows $f'(\alpha) > 0$" — citing "$\sqrt{h} < 1$" alone does NOT imply $f' > 0$. The actual condition is $\alpha^2(L^2 - m^2) > 0$.	▲Err	Minor	✓	Lack of clarity
E04c	Identifies $\lim_\ell Z^{[\ell]} = Z^*$ implicitly; FBS convergence needed to pass the bound to the limit.	▲Err	Minor	✓	Implicit assumption
E07	ReLU's σ' is undefined at 0; IFT used in smooth-derivative form via D without addressing the measure-zero non-differentiability.	▲Err	Minor	✓	Misapplication
E08a	Phase 1 worry: appendix sign mishandling. **Verified:** the equality $-[\mathbf{z}^\top \otimes J]P_A = J[\dots](I+K)(I \otimes A^\top)$ is correct.	✔False	–		
E08b	Appendix proves single-sample case; batch lift to N-sample matrix form (L461–L473) is stated without separate proof.	▲Err	Minor	✓	Lack of rigor
E09	L452: "Invertibility [of Q] is guaranteed by the strong monotonicity of $I - W$" — bare assertion. Needs active-subspace restriction.	▲Err	Major	✓	Gap in logics
E10	$\partial\hat{\mathbf{y}}/\partial\,\mathrm{vec}[Z^*] = \mathbf{u}^\top \otimes I_N$ is correct only under **row-major** vec, off by $K^{(n,N)}$ under standard column-major.	▲Err	Major	✓	Algebraic error
E11a	L1167: "Take the limit $\mu \to 0^+$, we have $\|\Pi\| \le 1$" — **false** for inactive ReLU coords ($J_{ii} = 0, \Pi_{ii} = 1/\mu \to \infty$).	▲Err	Critical	✓	Errors of omission
E11b	L1137: "$J_{ii} < 1\ \forall i \in [n]$" — for ReLU $J_{ii} \in \{0,1\}$, so $= 1$ in the active state, not < 1. Claim is literally wrong.	▲Err	Major	✓	Typo
E11c	$R_G = (I+G)^{-1}$ for G not verified maximal monotone; firmly-non-expansive bound on R_G not justified.	▲Err	Critical	✓	Gap in logics
E11d	L1183: 'typo (should be '$\times$').	▲Err	Minor	✓	Typo
E12	Inherits unproven gradient bounds from $\|(\mathbf{u}^\top \otimes I_N)Q^{-1}\| \le \|\mathbf{u}\|/m$.	▲Err	Critical	✓	Inherited Error
E13a	L595: "$(I-W)(Z-Z') = (W-W')Z' + (U-U')X +$ (ReLU correction)" with literal placeholder. Correct proof requires firm non-expansiveness.	▲Err	Critical	✓	Gap in logics
E13b	Mismatch in θ definition ($W, U, \mathbf{u}$ vs $A, B, U, \mathbf{u}$); reduction to A, B deferred to main theorem.	▲Err	Minor	–	Ambiguity
E15	(C1b) stated but never derived from drift. **Verified:** constants do match requirements exactly.	▲Err	Minor	✓	Lack of rigor
E16a	Phase-1 worry: dropped quadratic. **Verified false alarm:** identity used is exact; the description is just misleading.	✔False	Minor		
E16b	L875: No calculation shown for $\|Z^*(T^*) - Z^*(0)\|_F < \lambda_0/4$; reader must do 6 lines of algebra.	▲Err	Major	✓	Lack of rigor
E16c	L877: "$\sigma_{\min} \ge \lambda_0 - \lambda_0/4$" math right, but formatting "$\lambda_0 3/4$" should be $3\lambda_0/4$.	▲Err	Minor	✓	Typo
E16d	L728: "width is $O(N)$" — proof needs $n \ge N$ for $\lambda_0 > 0$. $O(N)$ is asymptotically misleading.	▲Err	Minor	✓	Missing assumption
E16e	(C1a), (C1b) never instantiated for any random init. Existential λ_0 lower bound never derived.	▲Err	Major	✓	Missing assumption
E16f	Abstract claims "gradient descent" rate; theorem is "Gradient Flow". Discrete-time GD result not closed.	▲Err	Major	✓	Unverified Claim
E17	Hypothesis stated on $[0, T)$ but integral on $[0, \infty)$. Standalone corollary is mis-scoped.	▲Err	Minor	✓	Lack of rigor
S01	Vec convention not stated; column-major vs row-major affects exact-formula correctness.	▲Err	Minor	✓	Ambiguity

Continued on next page

ID	Full Error Description	Status	Severity	Ver.	Type of Error
S02	"Strong monotonicity" used ambiguously for both PSD and operator norm interpretations.	▲Err	Minor	✓	Ambiguity

C. Auditing Example 2

We evaluate the framework on a real-world theoretical machine learning paper, titled Pruning at Initialisation through the lens of Graphon Limit: Convergence, Expressivity, and Generalisation (Pham et al., 2026), containing naturally occurring proof errors. After confirming with authors, we verified that our framework successfully generate a directed graph of the proof structure and mathematical issues, including missing assumptions, incorrect operator-theoretic arguments, unjustified limiting procedures, and inherited dependency errors.

C.1. Structure Graph

Figure 2 is the proof structure generated by our framework. We also provide a generated prose summary in the following for reference.

The paper builds a four-tier theoretical structure:

Tier 1 (Foundational): Assumptions A1–A5. Five regularity assumptions on the factorised saliency model

$$(S_n)_{ij} = \varphi_{n,i}\psi_{n,j}|\xi_{n,ij}|.$$

- **A1** is a mild growth condition on (d, n).
- **A2** demands the input features track a bounded continuous deterministic profile φ uniformly.
- **A3** demands the empirical CDF of neuron features converges uniformly to a deterministic CDF.
- **A4** demands the edge noises $\xi_{n,ij}$ are conditionally i.i.d. with continuous magnitude CDF, given the features and threshold.
- **A5** demands the threshold converges to a positive deterministic limit and is measurable with respect to the features σ-field.

Tier 2 (Main convergence theorem, Thm. 3.6). The main result

$$\delta_{\square}^{\mathrm{bip}}(W_n, \mathcal{W}) \to 0$$

is proved by triangle inequality:

1. **Concentration:**
$$\|W_n^{\pi} - \bar{W}_n^{\pi}\|_{\square} \to 0$$
via matrix-Bernstein (Lem. 09) using A1, A4 and the chain
$$\text{cut norm} \leftrightarrow \text{matrix cut norm} \leftrightarrow \text{operator norm}$$
(Lems. 06, 08).
2. **Mean-kernel L^1 convergence:**
$$\|\bar{W}_n^{\pi} - \mathcal{W}\|_{\square} \to 0$$
via continuous mapping + DCT (Lem. 11) using A2, A3, A4, A5 and order-statistic convergence (Lem. 10).
3. **Invariance** under row/column permutation (Lem. 07) closes the loop to bipartite cut distance.

Tier 3 (Verifications and extensions).

- Appendix B verifies A1–A5 for SNIP under bounded i.i.d. inputs and Gaussian initialisation. The verification has notable gaps in:
  - A3 (pointwise vs. uniform CDF),
  - A4 (asymptotic-not-exact independence),
  - A5 (top-ρ measurability).
- Appendix C shows GraSP score is asymptotically rank-equivalent to SNIP:
  - Case 1: piecewise-linear σ is exact,
  - Case 2: smooth σ with $o_p(1)$ remainder.
- Appendix D extends Thm. 3.6 layer-by-layer to depth-L networks via a union bound, using post-activation i.i.d. structure.

Tier 4 (Downstream consequences).

- **Expressivity (UAT).** With an "active rectangle" assumption (`A_uat_active_rect`) and a growth condition (`A_uat_growth`), Lem. 4.5 (dense core) uses a Chernoff bound on conditionally independent Bernoulli columns to find a

$$k \times \tilde{n}$$

fully connected subnetwork. Combining with Cybenko's classical UAT (Thm. 4.1) gives universality on active coordinates (Thm. 4.6).

- **Generalisation (Thm. 5).** Under a Graphon-NTK lazy-training assumption (`A_lazy_training`), apply Cao–Gu's NTK generalisation bound with the graphon-NTK Gram matrix $K_{\mathcal{W}}$.

- **Path density proposition (Prop. 5.2).** For $L = 2$, bound

$$|K_{\mathcal{W}}|$$

in terms of input correlations and the triple-graphon path density

$$P_{\mathcal{W}} = \int \mathcal{W}^{(1)}\mathcal{W}^{(2)}\mathcal{W}^{(3)}$$

using Gaussian covariance contraction lemmas (Lems. 26, 27).

C.2. Detected Errors

C.2.1. E01-A3-POINTWISE

'snip_case.tex' L467-474 explicitly writes "$\hat{F}_{\psi,n}(t) \to F_\psi(t)$ at continuity points t of F_ψ" — pointwise, not uniform. A3 demands uniform sup-norm convergence. Standard fix: invoke a Glivenko–Cantelli-type theorem for triangular arrays of conditionally-i.i.d. variables, which requires (i) the limiting CDF is continuous and (ii) uniform integrability. The paper does NOT establish (ii).

C.2.2. E02-A4-ASYMPTOTIC

`snip_case.tex`: Line 495 explicitly states "exact independence between $\xi_{n,ij}$ and $\psi_{n,j}$ does not hold at finite d". The "asymptotic decoupling" via $\mathrm{Corr} = O(d^{-1/2})$ (lines 500-509) gives a HEURISTIC, not a verification of A4. A4 is a non-asymptotic conditional-independence hypothesis. Theorem 3.6 applies to a SURROGATE model with literal independence; the bridge to literal SNIP is not closed.

C.2.3. E03-A5-MEASURABLE

`snip_case.tex`: L528: "in our factorised setting follows from the conditional i.i.d. structure of $\{S_{ij}\}$ given the row and column features (A4)". For SNIP, $S_{ij} = \varphi_i\psi_j|\theta_{ij}|$ — the scores include $|\theta_{ij}|$, so conditional i.i.d. given (φ, ψ) requires $\theta_{ij} \perp\!\!\!\perp \psi_j$, which fails (E02). Therefore the empirical CDF concentration ($\sup_t |\hat{F}_S - F_S| \to 0$) used to deduce $\hat{\tau}_n \to \tau$ rests on the same questionable independence.

C.2.4. E13-C1-NOT-CONSTANT

$|\dot{\Sigma}^{(1)}| \le H^2|\widetilde{\Sigma}^{(1)}| + c_1$ where "$c_1 = |[\sigma'(z(u_1, x))] \cdot [\sigma'(z(u_1, x'))]|$" is NOT a constant — it depends on u_1, x, x'. The next steps treat c_1 as a constant absorbed into C_1', which is INCORRECT. The dependence of c_1 on u_1, x, x' should be tracked.

D. Reproducibility

We implement our auditing pipeline using Claude Code (Anthropic, 2025) (built on Claude Opus 4.7) and execute it directly on the LaTeX source directories. A complete audit of the examples provided herein costs approximately $5 USD per run via the Claude API platform.

We provide the following contents of `SKILL.md` to ensure reproducibility with Claude Code.

```
---
name: proof-audit
description: Systematically audit a mathematical paper's proofs. Creates per-item audit
    files, tracks errors with status, and validates each error against the LaTeX source.
    Use when the user wants to audit a paper, verify proofs rigorously, check for
    mathematical errors, or reproduce corrected proofs for a blog or notes.
user-invocable: true
```

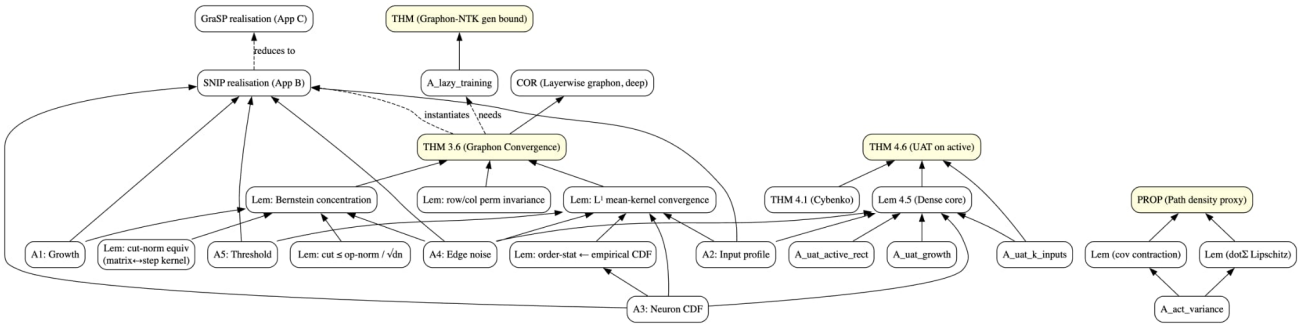

Figure 2. Dependency graph of the mathematical proof in Pham et al. (2026) showing the hierarchical structure of assumptions, intermediate lemmas, and main results.

```
allowed-tools: Read, Write, Edit, Bash, Glob
---

# /proof-audit     Mathematical Paper Proof Auditor

Arguments: `$ARGUMENTS`

Audit the proofs in a LaTeX paper by creating a structured `audit/` folder.
Run in three phases; which phases to run depends on the state of the repo and any argument
     the user passed.

---

## Dispatch on arguments and state

| Condition | Action |
|-----------|--------|
| No `audit/` folder yet | Run **Phase 1**, then **Phase 2** |
| `audit/` exists but errors are "    To validate" | Run **Phase 3** (Validation) |
| `audit/` exists and all errors confirmed | Report final summary to user |
| Argument is a `.tex` path | Use that file as the paper source |
| Argument is `validate` | Skip to Phase 3 only |
| No `.tex` file found | Ask the user which file to audit |

*Tool Hint:* To find the paper, use `Bash`: `find . -name "*.tex" -not -path "*/
    node_modules/*"`. If multiple `.tex` files exist, ask the user to pick one.

---

## Phase 1     Building the Audit Backbone

Scan the document to reconstruct the p a p e r s structure. The objective is to list all
    definitions, assumptions, lemmas, theorems, and corollaries, and establish their
    dependencies.

### 1a. Error Index
Create a file named `audit/error_table.md` containing a Markdown table:
`File | Error ID | One-line description | Status | Severity | Confidence`
*(Initially empty; to be populated in later steps).*

*   **Status:** `    To validate` | `       Confirmed Error` | `       False Alarm`
*   **Severity:** `Critical` | `Major` | `Minor`
*   **Confidence:**
    *   `High` (explicit contradiction, failed algebra, dimension mismatch)
    *   `Medium` (likely unstated assumption, suspicious leap, citation mismatch)
    *   `Low` (stylistic rigor concerns, ambiguous notation)

### 1b. Mathematical Item Files
```

Create one file per mathematical item (definition | assumption | lemma | theorem) in the order they appear. Number sequentially: `01_def_fbs.md`, `02_lem_grad_loss.md`, etc.

Format for each file:
```markdown
# NN. Type     Human Name (short_identifier)

**Source:** Paper section / label / approximate line numbers
**Item Status:**        Valid |        Warning (Needs Review) |        Contains Fatal Error

---
## Statement
(Full mathematical statement with MathJax: $$...$$)

## Original Proof
(Copy the original proof verbatim. Annotate references to other files using their short identifiers. Include contextual explanations.)

## Notes / Errors
(Initially empty. Record suspected gaps or errors here.)

```

1c. Proof Dependency Graph

Write `audit/structure.md` containing:

1. **Graphviz Output:** A ```dot` code block representing a dependency graph (Node A -> Node B if B depends on A).
2. **Prose Summary:** A layered explanation of how foundational elements build up to the main result.
3. **Structural Issues:** List missing assumptions, circular dependencies, or unproven regularity conditions.

Actionable: Record any structural issues found here in `audit/error_table.md` with status `    To validate`.

Phase 2   Line-by-Line Audit

Process each item file iteratively, starting from foundational items and following the dependency graph. Insert the following section after `## Original Proof`:

```markdown
## Audit Proof

(Rewrite the proof into a sequence of **atomic, verifiable steps**. Every step must be minimal, logically justified, and explicitly reference assumptions.)

| Step ID | Claim | Premises | Justification |
|---------|-------|----------|---------------|
| S1      | ...   | ...      | ...           |

```

Audit Imperatives:

* **Deconstruct:** Expand the proof. If multiple inferences happen in one sentence, split them into multiple `S`-steps.
* **Verify Variables & Conditions:** Highlight the introduction of new objects. If a theorem is invoked, explicitly verify its conditions in a dedicated step.
* **Flag Aggressively:** Actively try to break the proof. Look for hidden assumption drift, boundary case failures, or silent quantifier changes.

```
* **Record Flags:** If a step cannot be justified, add a note below the table. Log every
    missing justification or questionable logic in `audit/error_table.md` as `    To
    validate`.

---

## Phase 3     Validation

For each `    To validate` error in `audit/error_table.md`, find the exact line in the
    LaTeX source.

1. **Locate:** Use `Read` or `grep -B 3 -A 3` (to account for LaTeX line breaks) to
    inspect the source.
2. **Validate:** Does the source actually contain the error? Or was it a misinterpretation
    ?
3. **Update Item File:** Change item status. Add a one-sentence verification note with the
     exact line number. Provide mathematical proof of the error (e.g., dimension check, L'
    H pital computation).
4. **Update Index:** Change status in `error_table.md` to `       Confirmed Error` or `
           False Alarm`.

---

## Global Style & Tool Rules

* **MathJax:** Use `$$...$$` for block, `$...$` for inline. Never use standard markdown
    for math.
* **Notation:** Use standard macros (`\operatorname{vec}`, `\otimes`, `\|\cdot\|_2`). End
    proofs with $\square$.
* **Rigor:** Verify intermediate steps explicitly. Do NOT accept "by standard arguments."
* **Tool Usage:** Save files iteratively using `Write` or `Edit`. Do not attempt to
    process the entire paper in a single context window without saving intermediate states
     to the `audit/` folder.
```

E. Prompt for LLM-as-judge baseline

```
You are a mathematical referee reviewing a paper submitted to a peer-reviewed mathematics
    journal. You are given the raw LaTeX source of the paper. Use the LaTeX source to
    inspect the paper carefully, including notation, definitions, equations, assumptions,
    theorem statements, proof structure, and references to lemmas/propositions/theorems.
    Pay close attention to label consistency, hidden assumptions, and logical dependencies
     across the manuscript. Your task is to identify mathematical issues that would
    require revision before publication. Focus only on issues of mathematical substance,
    including but not limited to:

* Incorrect calculations, derivations, or algebraic manipulations
* Invalid or incomplete proofs
* Missing assumptions or unjustified inference steps
* Misuse or misstatement of known results
* Circular reasoning or logical gaps
* Incorrect asymptotic or probabilistic arguments
* Ambiguous or inconsistent notation affecting correctness
* Errors propagated from cited literature
* Statements that are false as written, overly general, or insufficiently justified
* Definitions that are incomplete or mathematically inconsistent
* Lack of rigor substantial enough to affect correctness Do not focus on purely stylistic,
     grammatical, or expository issues unless they obscure or invalidate the mathematics.
    For each issue you identify, provide:

1. A concise error identifier
2. The precise location in the paper
3. Relevant surrounding context
4. A detailed explanation of the issue and why it matters Output your final answer using
    the following Markdown format:
```

```markdown
## <Error ID>     <Short Name>

**Location:** <section/theorem/lemma/proof/equation label/page>

**Context:**
<quote or summarize the relevant mathematical statement or argument>

**Issue:**
<detailed explanation of the mathematical problem, missing assumption, invalid step,
    contradiction, or ambiguity>

**Why this matters:**
<explain the mathematical consequence or why revision is required>

```