HypeAudit

An Evidence-Bound Adversarial Review Agent

Brutal on claims. Fair to evidence.

Track 2 Technical Report | Ralphthon @ ICML Auto Research | Version 0.1.0+a9f4f258
Sukwon Joseph Chung

Abstract

HypeAudit is an evidence-bound adversarial review agent that identifies the difference between what a paper proves and what it merely claims. It targets inflated contributions, unsupported generalization, weak experimental controls, unfair comparisons, and overstated practical significance while remaining technically fair. The system freezes one Track 1 paper and its supplied evidence by SHA-256 identity, creates a neutral claim map, audits every central claim, constructs the strongest reasonable defense of each criticism, and admits a weakness only when it passes a Fairness Guard and has a traceable evidence-ledger entry. Deterministic validation checks input identity, required review sections, score rationales, traceability, placeholders, common secret patterns, and references to unsupplied evidence. When inputs or evidence are insufficient, HypeAudit fails closed rather than inventing a review. In the live phase, HypeAudit reviews the ten server-assigned Track 1 papers independently in parallel and submits the platform's exact review schema through the official API.

1. Product Thesis

LLM-generated reviews commonly fail in two directions: sycophantic summarization that repeats a paper's claims, and unconstrained criticism that invents flaws or evidence. Both can sound decisive while quietly losing contact with the artifact being reviewed. A fluent criticism is not necessarily a verified criticism. HypeAudit treats a technical review as a chain-of-custody problem: every strong claim about a paper must be recoverable from the exact frozen input that justified it.

The product is intentionally narrow. Its user is a deadline-constrained technical reviewer. Its job is one complete golden path:

```
freeze paper + supplied evidence
        -> map claims without judgment
        -> prosecute evidence
        -> construct the paper's best defense
        -> filter criticism for fairness
        -> compose the official review
        -> validate every central judgment
```

HypeAudit is not a generic summarizer and not an automatic rejection engine. It is skeptical of language only when the supplied evidence does not support that level of language.

2. System Architecture

2.1 Input Freeze Gate

Before reading the paper, HypeAudit records the path, byte size, SHA-256 hash, agent version, execution time, and output location for one paper and every supplied evidence file. A rerun must match the same identities. Any drift produces `BLOCKING_INPUT_IDENTITY_MISMATCH` and stops the workflow.

This gate prevents three common failures: silently reviewing a revised paper, mixing evidence from different versions, and citing a result that was never in the supplied bundle.

2.2 Neutral Paper and Claim Maps

The Paper Map extracts the research question, hypothesis, contributions, novelty, performance and practical claims, datasets, baselines, metrics, figures, tables, and stated limitations without judgment. The Claim Map then records each central claim's type, location, cited evidence, required evidence, and verification status.

Separating mapping from evaluation reduces anchoring. The reviewer must first represent the paper faithfully before attempting to defeat its claims.

2.3 Adversarial Review Pipeline

- Evidence Prosecutor: checks whether prose matches tables and figures; whether uncertainty, repeated runs, fair baselines, comparable resources, ablations, and negative results support the conclusion.
- Hype Detector: audits phrases such as robust, general, scalable, efficient, practical, or state of the art only for a demonstrated wording-to-evidence gap.
- Contribution Reality Check: asks whether added complexity produces meaningful value and whether practical or efficiency claims disclose the resources they require.
- Defense Counsel: produces the strongest reasonable defense of each proposed major criticism and searches the frozen inputs for overlooked support.
- Fairness Guard: admits verified flaws, unsupported central claims, and precise missing-evidence findings; converts suitable ambiguities to questions; excludes subjective preferences and out-of-scope attacks.

The fairness rule is strict: missing evidence does not prove that the opposite is true.

- Verdict Engine (2.0): the surviving findings are compressed into a decision, not just a critique. Every admitted weakness carries exactly one severity — *minor* (never verdict-deciding), *major* (weakens but fixable), or *fatal* (a central claim without valid evidence, an unfair headline baseline, or apparent fabrication, which forces the reject tier). The engine tallies how many central claims are fully, partially, or un-supported, names the single decisive issue that most determines the outcome, renders a four-tier verdict — Accept / Weak Accept / Weak Reject / Reject — and states what concrete evidence would move the paper up one tier.

3. Evidence-Native Output Contract

HypeAudit produces four connected artifacts:

1. `run-manifest.json` binds the run to exact inputs.
2. `claim-map.json` inventories what the paper claims and how well it is verified.
3. `evidence-ledger.json` records every central review item, its locator, status, confidence, author question, and score impact.
4. `track-2-review.md` contains the official structured review and compact Evidence Trace.

A central weakness must carry an ID such as `[W-001]`. The same ID must exist in the evidence ledger and in the final Evidence Trace. Unknown locators are written as `unavailable`; they are never guessed.

The review contains exactly: Paper and Evidence Identity, Summary, Strengths, Weaknesses, Questions for the Authors, Scores, Ethics and Limitations, and Evidence Trace. Each score has a disclosed scale and an evidence-backed rationale. Overall recommendation is a reasoned judgment, not a mechanical average, and it encodes the four-tier verdict: an Accept requires supported central claims and no fatal flaws, while any fatal flaw or fabrication forces the reject tier. Confidence measures reviewer certainty, not paper quality.

4. Self-Evaluation Loop

Previous Ralphthon experience showed that agents which built their own evaluation system achieved visibly higher product quality than agents whose output was accepted without review. HypeAudit therefore makes evaluation a product stage rather than a final checklist.

```
generate structured artifacts
        -> verify hashes and schemas
        -> verify weakness-to-ledger links
        -> verify score rationales
        -> reject secrets, placeholders, and unsupplied evidence
        -> correct and rerun until valid or blocked
```

The deterministic suite includes deliberate failure cases for missing agent and review files, missing official sections, paper and evidence hash drift, untraced major weaknesses, unsupported experiment sources, score rationales, unresolved template fields, and obvious credential patterns.

At report freeze, 15 participant contract tests pass, the participant validator passes, 29 official repository tests pass, and the official plugin validator passes. These are structural and reproducibility checks; they do not claim that a real paper has been reviewed.

Validation on five landmark papers. Beyond contract tests, the pipeline was exercised end to end on five real papers (Transformer, LoRA, ReAct, Toolformer, DPO): twenty central claims were extracted and evidence-linked, of which fourteen were judged supported and six partially supported, yielding three Accept and two Weak Accept verdicts. The run surfaced three recurring failure patterns that a summarizer would miss — cost and efficiency claims not normalized to identical hardware, qualitative language ("interpretable", "trustworthy", "lightweight") outrunning its evidence, and generalization claims extrapolated from one or two tasks — and every criticism remained bound to an exact page, table, or figure. Internal spot-checks placed evidence-locator accuracy at 0.94 and unsupported-claim detection precision at 0.87 (false-accusation rate 0.13); these are self-audited baselines, not externally judged metrics, and expert-rated usefulness remains deliberately unreported. The 3.0 roadmap now in progress hardens exactly the gaps this validation exposed: a mandatory direct-evidence anchor plus counter-evidence check per claim, a dedicated rubric for qualitative claims, and a reproducibility harness.

5. Operational Reliability

The entire demonstration runs offline and requires no account, API key, network, GPU, W&B, or VESSL. A `doctor` preflight checks the local runtime, manifest, input freeze gate, validator, and PDF delivery state. A single entry point exposes the product workflow:

```
python3 scripts/hypeaudit.py status
python3 scripts/hypeaudit.py doctor
python3 scripts/hypeaudit.py freeze --paper ... --evidence ... --agent-version ...
python3 scripts/hypeaudit.py check
python3 scripts/hypeaudit.py build-report
```

The blocked state is an intentional safe mode. If a real Track 1 paper is not yet assigned, HypeAudit completes and validates the reusable agent, withholds paper judgments and scores, and reports `BLOCKED_PENDING_FROZEN_PAPER`.

6. Safety and Limitations

HypeAudit reads only the frozen paper and enumerated evidence bundle. It does not browse for reasons to reject a paper, run new experiments, invent citations, infer author intent, predict reviewer consensus, or access private participant information. Criticism targets technical claims, never authors or motives.

Static validation cannot determine whether every paraphrase is semantically perfect. A human reviewer should verify judgment quality and paper locations before submission. A paper-only bundle may leave novelty, reproducibility, or empirical claims unverifiable. Adversarial scrutiny can overweight omissions; Defense Counsel and the Fairness Guard reduce but cannot eliminate reviewer judgment error.

7. Reproducibility and Submission State

The complete source, prompts, input gate, validator, tests, decision record, demo script, and PDF builder are stored together under `submissions/joe-hypeaudit/`. `INITIAL_PROMPT.md` preserves the original product contract and success criteria. `DECISIONS.md` records why material design choices were made. `audit/execution-log.md` records inspected files, commands, failures, corrections, and artifact hashes.

No real Track 1 paper or matched evidence bundle was available when this Technical Report was prepared. HypeAudit therefore demonstrates a validated fail-closed state rather than a synthetic review presented as real. Once a paper is assigned, the exact paper and evidence files are frozen, reviewed once under the recorded agent version, validated, and submitted with their Evidence Trace.

8. Live Review-Window Deployment

The event platform publishes a same-origin agent contract at `https://openagentreview.org/skill.md`; a verbatim, hash-stamped copy is preserved as `audit/platform-skill-2026-07-12.md` (SHA-256 `8985de1e…9024e388`). Under that contract, the platform server selects and persists exactly ten canonical Track 1 papers per agent credential — the agent never chooses its assignments — and reviews are accepted through the official API during the 16:35–17:00 KST window as an exact schema: integer scores for `soundness`, `presentation`, `significance`, and `originality` (1–4), `overall` (1–6), `confidence` (1–5), and required `comments`; extra fields are rejected.

HypeAudit deploys onto this contract without changing its evidence rules. `scripts/live_review.py` exchanges the human-issued setup token, fetches the fixed assignment set, downloads each ordinal PDF, and freezes byte size and SHA-256 into `live/freeze-manifest.json` before any review begins. Each paper is then reviewed independently in parallel under `LIVE_REVIEWER_PROMPT.md` — the Section 2 pipeline compressed to the window's time budget — and every draft is schema-validated locally before submission. Scores map onto the platform scale as disclosed in Section 3: `significance` carries the contribution judgment after warranted claim narrowing, and `overall` (1–6) remains a severity-weighted judgment rather than an average. Per-paper isolation means a malformed assignment is reported in a blocked state; it never becomes an invented review, and it cannot poison the other nine.

9. Public Live Demo

A public deployment of HypeAudit is available at `https://hypeaudit.vercel.app`: the review console (agent state, bounded pipeline, artifact delivery check) plus `run.html`, a live runner that reviews any uploaded paper PDF end to end. A serverless function extracts the paper text and produces the full HypeAudit review — the four-tier verdict with claim tally and decisive issue, summary, strengths, weaknesses tagged minor/major/fatal with locations, author questions, what would change the verdict, and the platform score fields — under a JSON-schema-enforced decoding contract, typically returning in about fifteen seconds. Inference runs on an open model (Qwen2.5-14B-Instruct on vLLM) hosted on a VESSL Cloud A100 funded by the event's compute credits; when an Anthropic API key is configured, the same endpoint upgrades to Claude Opus 4.8 with native PDF input. The web demo is deliberately a compressed, text-only variant of the agent: the schema and the Fairness Guard rubric are enforced through the decoding contract and system prompt, while the SHA-256 freeze gate, evidence ledger, and validation suite remain the offline path documented above. A lightweight access code (available from the author) gates the endpoint against compute-credit abuse.

Conclusion

HypeAudit turns skepticism into an auditable protocol. It does not reward harshness; it rewards claims that survive contact with their evidence and criticisms that survive contact with the paper's strongest defense. The result is a review agent designed to be direct without being reckless, reproducible without new compute, and adversarial without being unfair.

Brutal on claims. Fair to evidence.