WHEN PASSING CHECKS ARE NOT ENOUGH: SEPARATING EVIDENCE REPORTS FROM OUTCOME JUDGMENTS IN CODING AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Coding agents can pass their own tests and still falsely claim completion. We distinguish reporting verification events accurately from judging whether those events justify confidence in success. We introduce an E/J completion interface and verdict training on agent-generated trajectories. Evidence reports (E) are supervised by execution records, while outcome verdicts (J) predict hidden evaluation outcomes. On SWE-bench Verified, an SFT baseline with this interface performs verification but remains overconfident in its verdicts. We train J on the baseline's trajectories with two targets: Hard-J uses observed outcomes, and Soft-J blends outcomes with a difficulty-conditioned estimate of evidence reliability. The two targets have complementary strengths. On fixed development histories, Hard-J raises AUC from 0.590 to 0.787, while Soft-J reduces expected calibration error from 0.294 to 0.083. On 72 held-out tasks, Hard-J makes the fewest false-positive verdicts after passing checks (28%, versus 39% for Soft-J and 83% for SFT), while Soft-J has lower calibration error and Brier score than independently calibrated Hard-J on every trajectory source, with comparable discrimination among passing-check episodes. Ablations and report interventions show that verdicts condition on reported evidence, and directly incentivizing positive self-reports induces reporting hallucinations, reinforcing the need to supervise process fidelity separately from outcome judgment.

1 INTRODUCTION

Language-model agents can navigate repositories, modify source code, and execute tests through dedicated software interfaces (Yang et al., 2024). Benchmarks such as SWE-bench evaluate these capabilities on real repository issues (Jimenez et al., 2024). Alongside the ability to produce a correct patch, reliable delegation requires an agent to judge whether its work is complete. Recent studies document false success claims in agent benchmarks (Advani, 2026) and inaccurate self-reporting in developer–agent interactions (Tang et al., 2026). A completion statement therefore needs to communicate how much confidence the available evidence warrants.

Research on uncertainty estimation offers several ways to improve such judgments. Predictive accuracy does not ensure calibrated confidence in neural classifiers (Guo et al., 2017) or language-model question answering (Jiang et al., 2021). Prompting can elicit calibrated verbal confidence on question-answering benchmarks (Tian et al., 2023), while supervised learning and reinforcement learning can teach models to express confidence aligned with empirical correctness (Lin et al., 2022; Xu et al., 2024; Damani et al., 2025). At the agent level, concurrent work predicts task success from internal trajectory and action representations (Mammen et al., 2026). We study how a coding agent can faithfully report its verification evidence and use it to express confidence in its own task outcome.

A coding agent helps choose the checks that it later uses to justify completion. Those checks reflect its interpretation of the issue and may omit requirements that the patch fails to satisfy. In function-level code generation, expanded test suites expose errors missed by the original tests (Liu et al., 2023). Consequently, even a truthful report that all executed tests passed can support an overconfident verdict. The reliability of a passing check may also vary with task difficulty. This gives two

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

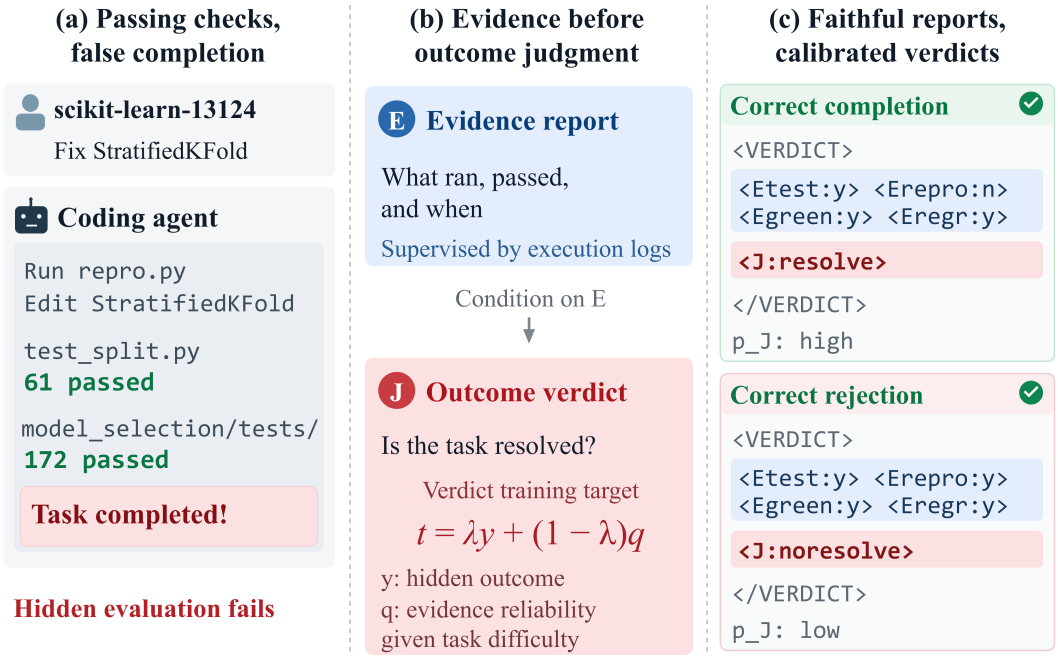


Figure 1: Overview. (a) Passing checks can precede false completion (adapted from Case B, Appendix F). (b) Execution records supervise E; J uses the target $t = \lambda y + (1 - \lambda)q$, with $\lambda \in [0, 1]$. Hard-J uses $\lambda = 1$; Soft-J blends outcomes with difficulty-conditioned evidence reliability. (c) Illustrative outputs: blue tokens report evidence, red tokens predict resolution, and p_J comes from J-token logits.

distinct learning problems: accurately reporting observable verification events, and interpreting their strength as evidence of success. We study this distinction through the question: *what did the agent check, and how strongly does that evidence support completion?*

We introduce an E/J completion interface (Figure 1) and verdict training on agent-generated trajectories. Execution records supervise process reports; hidden outcomes supervise completion judgments, with hidden tests unavailable at inference. An SFT baseline already uses this interface and learns to verify, but remains overconfident. Controlled report interventions show that its verdict relies heavily on positive evidence claims.

We then train verdicts on fixed trajectories collected from that baseline. *Hard-J* uses observed outcomes as targets; *difficulty-conditioned Soft-J* blends outcomes with estimated evidence reliability. Because SFT already uses the interface, their gains over SFT isolate the effect of verdict training. On fixed development contexts, Hard-J raises AUC from 0.590 to 0.787, and Soft-J reduces expected calibration error (ECE) from 0.294 to 0.083. On 72 held-out tasks, we cross-score fixed trajectories from all three policies. Soft-J has lower ECE and Brier score than independently calibrated Hard-J for every trajectory source, with comparable Green AUC (pooled 0.678 versus 0.690). Both reduce generated false-positive verdicts after passing checks, from 83.3% for SFT to 28.0% for Hard-J and 39.3% for Soft-J.

Our contributions are as follows:

- We introduce an E/J completion interface that supervises process reports and outcome judgments separately (§4).
- We train verdicts on agent-generated trajectories with two targets: observed outcomes (Hard-J) and a difficulty-conditioned blend of outcomes and evidence reliability (Soft-J) (§4.3).
- We evaluate fixed histories, new rollouts, and cross-scored trajectories to distinguish calibration from discrimination, and use report interventions and incentive experiments to examine evidence dependence and reporting fidelity (§6.3, §6.4, §7).

2 Related Work

Learning and eliciting confidence. Lin et al. (2022) supervise verbalized confidence with empirical accuracy; Kapoor et al. (2024) fine-tune correctness estimates, including a two-token LoRA formulation. Self-REF learns confidence tokens following the model's own answer (Chuang et al., 2025). SaySelf learns confidence and self-reflective rationales without inference-time resampling (Xu et al., 2024); RLCR rewards correctness and Brier score (Damani et al., 2025). Kadavath et al. (2022) study answer-conditioned $P(\mathrm{True})$ and question-level $P(\mathrm{IK})$. Prompted verbal confidence (Tian et al., 2023) and semantic entropy over sampled outputs (Kuhn et al., 2023) provide other estimates. At deployment, our acting model reports evidence E and generates a terminal verdict J after a single task attempt, without a separate verifier or additional candidate rollouts.

Agent success prediction and monitoring. Reflexion uses verbal self-reflection and memory for repeated trials (Shinn et al., 2023), while SWE-Gym trains trajectory verifiers for best-of-N selection (Pan et al., 2025). SWE-RM studies reward-model discrimination and calibration (Shum et al., 2025); HTC fits a separate calibrator to trajectory features (Zhang et al., 2026). Concurrent work predicts success from internal action representations (Mammen et al., 2026). False success claims occur in benchmarks and developer sessions (Advani, 2026; Tang et al., 2026). Guan et al. (2025) categorize monitorability signals; we use fixed false-positive budgets (METR, 2026) to evaluate failure detection from the agent's verdict confidence.

Process supervision, fidelity, and software evaluation. Process supervision scores intermediate reasoning (Lightman et al., 2024); E checks recorded execution events. Confessions rewards report honesty separately from the main answer (Joglekar et al., 2025). Explanation faithfulness concerns whether rationales reflect internal computation (Jacovi & Goldberg, 2020; Turpin et al., 2023); our reports are checked against execution logs. J targets resolution on SWE-bench Verified, determined by held-out fail-to-pass and pass-to-pass tests (Jimenez et al., 2024; Chowdhury et al., 2024). Test scope bounds this label: Verified retains tasks that annotators judged well specified with appropriately scoped tests (Chowdhury et al., 2024), yet a PatchDiff-based audit still finds incorrect test-passing patches (Wang et al., 2026).

3 Motivating the E/J Separation

An initial diagnosis on 500 SWE-bench Verified issues (Chowdhury et al., 2024) distinguishes missing verification from failure despite passing checks. Among detected false completion claims, 79.5–88.3% for GLM-5.2 (GLM-5-Team, 2026) follow a passing post-edit check, whereas 76.5% for Qwen3-32B (Yang et al., 2025) involve no recognized check. Prompting GLM with testing skills increases checking, but task success and false claims per episode change little; their confidence intervals include no change. Appendix B gives the protocols, complete figures, and counts. These findings motivate separate supervision for observable verification reports and outcome judgments: neither a success label nor a passing check establishes both. Our subsequent experiments evaluate report fidelity and verdict learning through the E/J interface.

4 Method: Separating Evidence Reporting from Outcome Judgment

4.1 E/J Interface

Dedicated tokens have been used for tool invocation (Hao et al., 2023), retrieval and self-critique (Asai et al., 2024), and failure-state signaling for structured recovery (Huang et al., 2026). Inspired by these structured interfaces, we use separate evidence and verdict tokens to distinguish auditable execution reports from outcome judgments, with supervision tailored to each.

For an issue x, let h denote the observable execution history and $y \in \{0, 1\}$ the hidden evaluation outcome. At completion, the agent emits four binary evidence fields $\widehat{E}$ followed by a binary verdict $\widehat{J}$:

```
<VERDICT> <Etest:y/n> <Erepro:y/n> <Egreen:y/n>
  <Eregr:y/n> <J:resolve/noresolve> </VERDICT>
```

Here `y/n` denotes a choice between the positive and negative tokens. E-first ordering allows the verdict to condition on the report as well as x and h.

The verdict probability is $p_J = \sigma(\ell_{\texttt{resolve}} - \ell_{\texttt{noresolve}})$, the restricted softmax over the two J-token logits.

4.2 Separate Supervision for Evidence and Judgment

An execution parser produces the reference $E^*(h)$: `Etest` records any recognized test or reproduction run; `Erepro` records a recognized reproduction command before the first source edit; `Egreen` records a passing check after the last source edit; `Eregr` records a check after the *first* source edit, providing a post-change-check flag. `Egreen` accepts any recognized passing check after the last edit; the diagnosis in Appendix B uses the last check's result. Appendix C gives the full parser definitions, including no-edit cases. Initial SFT uses token cross-entropy on agent actions and report tokens, with E targets from these records and J targets from y; Hard-J and Soft-J then continue training J (§4.3). E thus reports observable execution events, while J predicts the hidden evaluation outcome.

4.3 Verdict Training on Agent-Generated Trajectories

We collect trajectories from the SFT baseline and continue training only at the verdict position. Both variants use the same fixed histories, generated E prefixes, and update budget. ***Hard-J*** uses observed outcomes directly ($t_r = y_r$). ***Soft-J*** targets calibration: it blends outcomes with an estimate of evidence reliability given task difficulty. Let E_r^* and y_r be the parser evidence and outcome of rollout r on issue i. We estimate evidence reliability using

$$q_r = \sigma\left[\operatorname{logit}(\widetilde{p}_{i,-r}) + b_{-f(i)}^{\top}(E_r^* - \overline{E}_{i,-r}^*)\right], \qquad t_r = \lambda y_r + (1-\lambda) q_r. \tag{1}$$

Here $\lambda \in [0,1]$: Hard-J uses $\lambda = 1$ and Soft-J uses $\lambda = 0.3$ in our experiments. Here $\widetilde{p}_{i,-r}$ is the success rate of other submitted rollouts of the same issue, shrunk toward the overall training success rate with two pseudo-observations; $\overline{E}_{i,-r}^*$ is their mean evidence vector. We fit b by regularized logistic regression with issue intercepts on the other folds of a five-fold issue split. Thus the target combines difficulty estimated from other rollouts of the same issue with an evidence adjustment centered at the mean evidence for those rollouts. The outcome mixture retains sample-specific supervision. The precise estimator, including prior estimation, is in Appendix D.1. Hidden tests, their outcomes, and q are unavailable at inference, which uses the issue, execution history, and generated E prefix. Empirical-accuracy targets also appear in APRICOT's auxiliary calibrator (Ulmer et al., 2024); combining hard labels with training-only soft targets relates to generalized distillation (Lopez-Paz et al., 2016).

Starting from the SFT baseline, both methods continue its LoRA adapter (Hu et al., 2022) using $\mathcal{L}_J = -t_r \log p_{J,r} - (1-t_r)\log(1-p_{J,r})$ only at the J position, with embeddings frozen. During training, the recorded trajectory and generated E prefix remain fixed; only the target differs, with parser-derived E supplying Soft-J's reliability estimate q. Training does not refresh trajectories as the adapter changes. Fresh rollouts evaluate how the shared adapter update affects acting behavior.

5 Experimental Setup

Models. Our SFT baseline is Qwen3-32B (Yang et al., 2025) trained on GLM-5.2 (GLM-5-Team, 2026) demonstrations collected with skills, SWE-Gym trajectories (Pan et al., 2025), and base-policy editing/reporting examples. Hard-J and Soft-J continue its LoRA adapter on fixed SFT-policy trajectories with matched verdict-only update budgets. Data provenance and training settings appear in Appendix D and Table 9. A post-hoc logistic calibrator fits the frozen J margin and parser-derived E fields on verdict-training contexts. We also evaluate Hard-J with an independent calibrator fitted on a separate calibration set, excluding both the fixed development contexts and the 72-task evaluation set. Prompted binary and verbalized confidence baselines use the frozen SFT model without additional training (Appendix D.1).

Table 1: SFT baseline behavior and report fidelity on 104 tasks. All table values are percentages. Behavior rates use all episodes; success given passing checks conditions on that event. Fidelity evaluates generated reports (51.6% coverage), and over-claim conditions on a positive reported field. Sample sizes and field-level fidelity statistics across task splits appear in Table 8.

Verification and task outcomes

Measure	Rate (%)
Own checks pass	33.0
Task success	14.7
Success given passing checks	34.5

Report fidelity

Field	Accuracy (%)	Over-claim (%)
`Etest`	85.7	6.0
`Egreen`	74.2	11.0
`Eregr`	83.5	11.0
`Erepro`	70.2	87.0

Data and splits. Both the 104-task analysis set and the 72-task evaluation set are issue-disjoint from the entire SFT corpus, including GLM demonstrations, SWE-Gym trajectories, and base-policy examples. The 104-task set supplies rollouts for behavior analysis and verdict learning; an issue-disjoint split of the verdict corpus yields training and development contexts. The 72-task set is also disjoint from the verdict-training and verdict-development sets and is not used for development tuning or checkpoint selection. It supports our primary new-rollout evaluation. Corpus composition, split details, and sample counts appear in Appendices D, D.1, and A.

Fixed-context evaluation. We hold execution histories and generated E prefixes fixed across models and read next-J-token probabilities via the restricted softmax. Primary token-model results use the full development set; prompted baselines, interface comparisons, and report interventions use its well-formed-report subset, with matched token-model results on that subset.

New-rollout evaluation. The SFT baseline, Hard-J, and Soft-J independently generate trajectories in mini-SWE-agent (Yang et al., 2024; Lieret & Jimenez, 2026) on the 72-task set. We evaluate emitted reports and verdicts ($\widehat{E}$, $\widehat{J}$), and separately score verdict probabilities (p_J) on each model's trajectories. Appendix A gives rollout settings and report counts; Appendix E covers complementary rollouts on the 104-task set, which overlaps the verdict-training corpus. We also cross-score each source's fixed completion contexts with all verdict models, holding the execution history and generated E prefix constant within each comparison. This separates scorer comparisons from differences in the trajectories each policy generates.

Metrics. We measure per-field accuracy $P(\widehat{E}_k = E^*_k)$, over-claim rate $P(E^*_k = 0 \mid \widehat{E}_k = 1)$, under-claim rate $P(\widehat{E}_k = 0 \mid E^*_k = 1)$, and report coverage. Among valid reports, the false-positive verdict rate on unresolved episodes with passing checks (the *failed-green* subset) is

$$\mathrm{FC}^{\mathrm{gen}}_{\mathrm{green}} = P(\widehat{J} = 1 \mid E^*_{\mathrm{green}} = 1,\ y = 0). \tag{2}$$

This differs from the natural-language claim detection in Appendix B. For p_J, we report Brier score, 10-bin equal-width ECE, overall ROC-AUC, AUC among passing-check episodes (green AUC), and within-issue ranking accuracy. The thresholded analog, $\mathrm{FC}^{p_J>0.5}_{\mathrm{green}}$, replaces $\widehat{J} = 1$ with $p_J > 0.5$ in Eq. 2, distinguishing probability thresholds from generated verdicts. All report-conditioned metrics condition on report emission.

6 Results

6.1 Verification Behavior and Report Fidelity

On the 104-task analysis set, the SFT baseline executes verification and improves task success over the unadapted model. Reports of test execution, passing checks, and post-edit checking show low over-claim rates (Table 1). Reports appear in about half of episodes, and some parser-verified passing checks go unreported. Appendix F presents qualitative case studies illustrating representative alignments and misalignments between evidence reporting and verdict correctness.

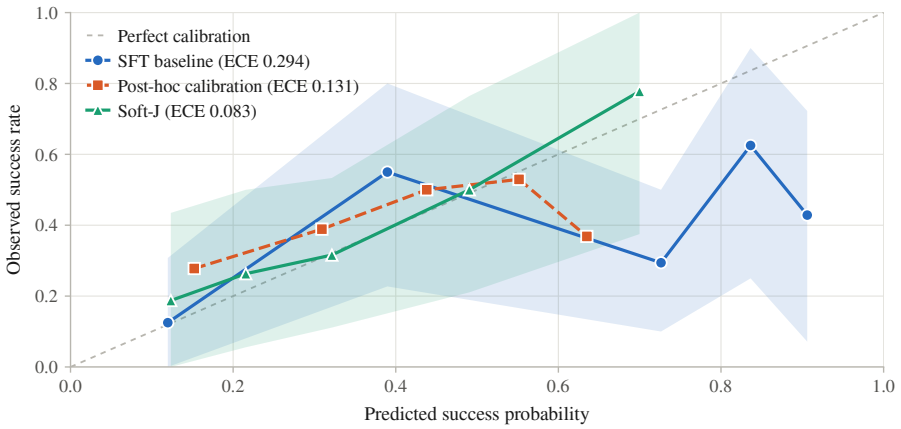


Figure 2: J reliability diagrams on fixed development contexts. Five equal-mass bins per series; shading denotes pointwise 95% issue-cluster bootstrap intervals. ECE is computed over ten equal-width bins. Appendix E details sample counts and uncertainty estimation.

The pre-edit reproduction field (`Erepro`) exhibits frequent over-claiming. This discrepancy aligns with sparse positive reproduction labels in the verdict-training corpus and the absence of a dedicated reproduction action token in the GLM source. Broadening the reproduction criteria reduces reporting accuracy rather than resolving the mismatch (Appendix D). Refitting the Soft-J reliability target without `Erepro` yields highly correlated targets with similar outcome discrimination (Appendix E.4).

6.2 Passing Checks Are Informative but Insufficient

Across the 104-task set, episodes with passing checks are much more likely to resolve the issue than those without them (Table 1). This association also holds within issues: when a resolved and an unresolved rollout differ in whether checks pass, the passing check usually belongs to the resolved rollout. Nevertheless, most episodes with passing checks still fail hidden benchmark evaluation. Differential line coverage provides weak separation between successful and failed passing-check rollouts (Appendix E.3), indicating that structural line execution alone does not imply semantic test sufficiency. A passing check therefore raises the probability of success but does not establish it.

6.3 The Verdict Over-Relies on Reported Evidence

To test whether the agent's verdict actively conditions on its emitted evidence, we conduct counterfactual interventions on fixed development contexts with well-formed reports (§5). Holding the trajectory constant and flipping a single reported field from negative to positive substantially inflates the predicted completion confidence p_J for test execution, passing checks, and post-edit checking (Table 14). The verdict therefore responds directly to reported evidence even when the execution history is unchanged.

6.4 Hard-J and Soft-J: Calibration and Discrimination

Fixed-context development results. Both verdict continuations improve discrimination over SFT on the same development contexts (Table 2). Hard-J reaches AUC 0.787 versus 0.730 for Soft-J, while Soft-J reduces ECE from 0.294 to 0.083. Independent calibration improves Hard-J's ECE to 0.108; it leads on overall AUC, Green AUC, and Brier score, while Soft-J leads on ECE and within-issue ranking. Prompted baselines remain overconfident after passing checks. Figure 2 shows reliability diagrams and bootstrap confidence bands.

New-rollout results. When each model scores its own new rollouts, Soft-J has the best probability metrics (Table 12). Because the trajectory populations differ, we also cross-score fixed trajectories below.

Table 2: Fixed-context development results. *Prompted baselines evaluate the subset with complete, correctly formatted reports. Sample counts and token-model comparisons on that subset appear in Appendix E. FC and within-issue ranking are percentages; ranking excludes ties. Lower Brier, ECE, and FC are better; higher AUC/ranking is better. Bold marks best values on the full context set.

Method	AUC	Brier	ECE	Green AUC	$FC_{green}^{p_J>0.5}$ (%)	Within-issue ranking (%)
SFT baseline	0.590	0.323	0.294	0.494	90	78
+ post-hoc calibration	0.569	0.251	0.131	0.457	72	75
Hard-J	**0.787**	0.211	0.208	**0.735**	**10**	86
Hard-J + independent calibration	**0.787**	**0.198**	0.108	**0.735**	17	86
Soft-J	0.730	0.205	**0.083**	0.676	17	**90**
Prompted $P(\text{True})^*$	0.577	0.508	0.515	0.554	100	86
Prompted verbalized*	0.596	0.535	0.547	0.518	100	100

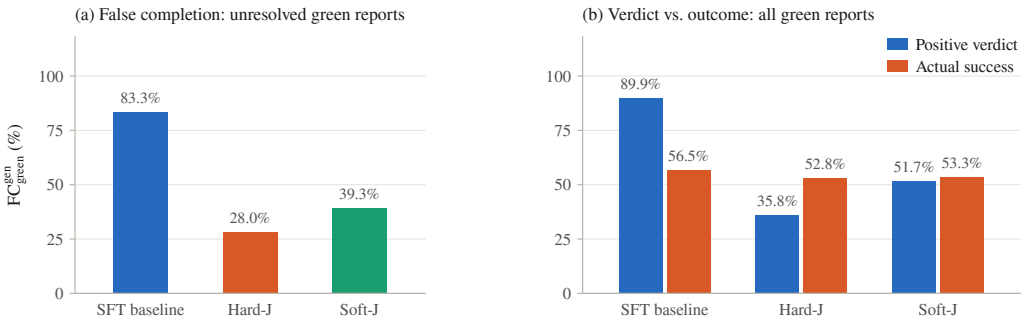


Figure 3: Generated completion declarations on new rollouts of 72 tasks held out from both SFT and verdict training. (a) False-claim rate among unresolved episodes with passing checks (FC_{green}^{gen}). (b) Verdict declaration and ground-truth success rates among all passing-check reports. Bars reflect distributions within each model's generated population. Hard-J uses hard-label-only verdict training. Appendix A gives counts and additional metrics.

For generated verdicts, both methods reduce the false-positive rate among unresolved green episodes (FC_{green}^{gen}): 83.3% for SFT, 28.0% for Hard-J, and 39.3% for Soft-J (Figure 3a). Hard-J makes fewer false-positive declarations but declares success less often than the outcome rate among green reports (36% versus 53%); Soft-J's declaration rate matches the outcome rate (52% versus 53%; Figure 3b). Task success and report coverage are similar across models (Appendix A). Missing reports in this evaluation arise predominantly from episodes that exhaust the 80-step budget (179 of 200 missing reports across the SFT baseline and Soft-J); none of these budget-exhausted episodes is resolved, and they contain no completion claim. Additional rollouts on the 104-task analysis set show a similar pattern (Appendix E).

Cross-scoring fixed held-out trajectories. With the trajectory source held fixed, Soft-J has lower ECE than independently calibrated Hard-J on all three sources (0.111–0.142 versus 0.158–0.171; Table 3) and lower Brier score, with the smallest gap on Hard-J trajectories (0.226 versus 0.228). Green AUC is comparable: Hard-J leads on SFT and Soft-J trajectories, Soft-J leads on Hard-J trajectories, and pooled values are 0.690 and 0.678 (Appendix E.2).

7 Ablations and Mechanism

What does the target contribute? To isolate the sources of calibration gain, we train LoRA continuations with matched update budgets comparing the hard outcome (y), soft evidence reliability (q), and task difficulty components (Table 4; training hyperparameters in Appendix D). The

Table 3: Cross-scoring on the 72-task held-out set. Within each row, scorers evaluate the same completion contexts. Hard+cal denotes independently calibrated Hard-J; its Green AUC matches uncalibrated Hard-J. Bold marks the better point estimate in each pair. Appendix E.2 gives all scorers and additional metrics.

	ECE ↓		Brier ↓		Green AUC ↑	
Trajectory source	Hard+cal	Soft-J	Hard+cal	Soft-J	Hard+cal	Soft-J
SFT	0.171	**0.118**	0.237	**0.212**	**0.688**	0.650
Soft-J	0.158	**0.111**	0.226	**0.199**	**0.719**	0.700
Hard-J	0.164	**0.142**	0.228	**0.226**	0.662	**0.681**

Table 4: Target and interface ablations on fixed development contexts; bold marks the best value in each column within each panel. All continuations initialize from the SFT baseline; interface rows evaluate the well-formed-report subset, where the default row is Soft-J. Lower ECE and FC are better; higher AUC is better. Sample counts appear in Table 10; secondary metrics appear in Table 11.

		AUC	ECE	Green AUC	$\mathrm{FC}_{\mathrm{green}}^{p_J>0.5}$ (%)
Training target					
Soft-J	$0.3y + 0.7q$	0.730	**0.083**	0.676	17
Hard label only	y	**0.787**	0.208	**0.735**	10
Hard label + post-hoc	y, then rescaling	0.786	0.211	**0.735**	**7**
Hard-J + independent calibration	y, then rescaling	**0.787**	0.108	**0.735**	17
Soft label only	q	0.717	0.098	0.670	21
Difficulty only	$0.3y + 0.7\tilde{p}_{i,-r}$	0.701	0.091	0.621	28
No difficulty	$0.3y + 0.7q_{\mathrm{pooled}}$	0.673	0.086	0.568	34
Report interface (well-formed-report subset)					
Default	Tokens, E before J	**0.729**	**0.076**	**0.688**	**14**
Natural language	Prose, E before J	0.707	0.155	0.593	25
J first	Tokens, J before E	0.684	0.106	0.621	25

difficulty-only variant replaces q with the leave-one-out task difficulty baseline while retaining the outcome mixture. The *no-difficulty* variant replaces q with q_{pooled}, an evidence-only success estimate derived from regularized logistic regression on the four parser-derived E fields pooled across training issues, omitting issue-specific difficulty baselines (Appendix D.1).

The Hard-J/Soft-J comparison in §6.4 motivates these component ablations. On fixed development contexts, hard-label supervision yields strong discrimination and underconfident probabilities. Independent calibration improves its probability estimates, while Soft-J achieves lower ECE (Table 4). Soft reliability targets provide strong calibration, and mixing in observed outcomes improves discrimination and within-issue ranking. Difficulty-only retains low ECE but weakens discrimination; the full target lowers false-positive verdicts after passing checks from 28% to 17%. Removing difficulty conditioning also weakens Green AUC and increases false-positive verdicts. These complementary ablations show the value of interpreting evidence relative to task difficulty.

Conditioning and format. Reversing generation order to emit J before E prevents the verdict token from autoregressively conditioning on reported process flags, weakening calibration and discrimination and increasing false-positive verdicts (Table 4). Under a matched continuation budget, structured token reporting also achieves sharper calibration than free-form natural language, despite comparable within-issue ranking. The counterfactual interventions of §6.3 show that Soft-J substantially reduces the confidence increase when the test, green, or regression field changes from negative to positive (Table 14). Soft-J thus discounts positive reports while retaining evidence-conditioned verdicts.

Rewarding the report. A separate experiment on verdict-development issues tests whether agents can be incentivized to verify by targeting self-reports directly versus reinforcing verified behaviors. We compare rollouts after (1) supervised fine-tuning that upweights successful trajectories with parser-verified checks while preserving factual report labels, versus (2) directly applying positive token rewards to E. The direct token incentive induces severe reporting hallucination: all emitted reports claim complete verification across all four fields ($E = (y, y, y, y)$) alongside positive verdicts, causing reproduction-report accuracy to collapse. Actual passing checks increase far less dramatically than the reports suggest. In contrast, trajectory reweighting preserves low over-claim rates on audited fields (Table 9 details budgets; full counts in Appendix E.5).

8 Auditability and Utility

For post-execution failure triage, we flag episodes by the failure-risk score $1 - p_J$ at nominal false-positive rates of 10% and 20% on successful rollouts (METR, 2026). Across all usable completion contexts, Soft-J catches more failures than SFT at both budgets on fixed contexts and new rollouts. Prompted confidence is less effective on the well-formed subset (Appendix E.6 describes threshold selection and reports catch rates with bootstrap intervals).

Because each E field has a parser-derived reference, users and runtime systems can audit individual evidence claims against execution logs independently of J.

9 Conclusion and Limitations

Faithful process reporting and calibrated outcome judgment are separate learning problems. Our approach combines an E/J interface with verdict training on agent-generated trajectories. Outcome-supervised Hard-J gives the strongest discrimination on development contexts, and difficulty-conditioned Soft-J gives the best calibration. On 72 SWE-bench Verified tasks held out from SFT and verdict training, Soft-J has lower ECE and Brier score than independently calibrated Hard-J on fixed trajectories from all three policies, with comparable Green AUC. Both reduce generated false-positive verdicts after passing checks relative to SFT, with similar task success. Direct incentives for positive reports induce reporting hallucinations, reinforcing the need to supervise process faithfulness against execution records.

Our experiments use Qwen3-32B and mini-SWE-agent on SWE-bench Verified, where parser-derived labels provide deterministic behavioral proxies for verification commands. Cross-scoring comparisons are point estimates without paired uncertainty intervals. Report-conditioned evaluation also excludes episodes without usable reports. Future work includes evaluation across additional model families and benchmarks, where calibration can vary under distribution shift (Ovadia et al., 2019), and prospective evaluation of verdict confidence in controllers for adaptive test generation, early stopping, and automated patch refinement.

AI USE STATEMENT

In this work, generative AI assisted with method implementation and data-processing code. We additionally used generative AI for grammar checking. Language models were also evaluated as software engineering agents, as described in §5 and the appendices.

The authors reviewed the AI-assisted text, pipeline scripts, and data-processing code. The authors take full responsibility for the manuscript and associated artifacts, including AI-assisted text and code.

ETHICS STATEMENT

Incorrect completion claims can lead developers to adopt flawed or untested patches. Calibrated verdicts support failure triage alongside integration testing and human code review. Our experiments cover non-adversarial task failures. All experimental artifacts, repository snapshots, and model weights adhere to the open-source licenses of their respective sources, preserve double-blind submission anonymity, and omit private credentials or sensitive execution environment traces.

REPRODUCIBILITY STATEMENT

The appendices document the experimental workflow: (1) **Evaluation setups and benchmark metrics:** The 72-task evaluation design and generation statistics are detailed in Appendix A, with operational metric formulations and parser definitions given in Appendices B and C. (2) **Data curation and training protocols:** Training split constructions, trajectory synthesis, SFT parameters, Hard-J training, and difficulty-conditioned Soft-J target fitting are described in Appendices D and D.1. (3) **Extended empirical results and analysis:** Appendix E provides full numerical tables, cross-scoring comparisons, and reliability diagrams, while Appendix E.6 documents retrospective monitoring threshold selection and bootstrap uncertainty calculations. All training scripts, evaluation harnesses, and parsed trajectory records will be released publicly upon publication.

REFERENCES

Laksh Advani. From confident closing to silent failure: Characterizing false success in LLM agents. *arXiv preprint arXiv:2606.09863*, 2026. doi: 10.48550/arXiv.2606.09863. URL https://arxiv.org/abs/2606.09863.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-RAG: Learning to retrieve, generate, and critique through self-reflection. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=hSyW5go0v8.

Neil Chowdhury, James Aung, Chan Jun Shern, Oliver Jaffe, Dane Sherburn, Giulio Starace, Evan Mays, Rachel Dias, Marwan Aljubeh, Mia Glaese, Carlos E. Jimenez, John Yang, Leyton Ho, Tejal Patwardhan, Kevin Liu, and Aleksander Madry. Introducing SWE-bench Verified. OpenAI blog, August 2024. URL https://openai.com/index/introducing-swe-bench-verified/.

Yu-Neng Chuang, Prathusha Kameswara Sarma, Parikshit Gopalan, John Boccio, Sara Bolouki, Xia Hu, and Helen Zhou. Learning to route LLMs with confidence tokens. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 10859–10878. PMLR, 2025. URL https://proceedings.mlr.press/v267/chuang25b.html.

Mehul Damani, Isha Puri, Stewart Slocum, Idan Shenfeld, Leshem Choshen, Yoon Kim, and Jacob Andreas. Beyond binary rewards: Training LMs to reason about their uncertainty. *arXiv preprint arXiv:2507.16806*, 2025. doi: 10.48550/arXiv.2507.16806. URL https://arxiv.org/abs/2507.16806.

GLM-5-Team. GLM-5: From vibe coding to agentic engineering. *arXiv preprint arXiv:2602.15763*, 2026. doi: 10.48550/arXiv.2602.15763. URL https://arxiv.org/abs/2602.15763.

Melody Y. Guan, Miles Wang, Micah Carroll, Zehao Dou, Annie Y. Wei, Marcus Williams, Benjamin Arnav, Joost Huizinga, Ian Kivlichan, Mia Glaese, Jakub Pachocki, and Bowen Baker. Monitoring monitorability. *arXiv preprint arXiv:2512.18311*, 2025. doi: 10.48550/arXiv.2512.18311. URL https://arxiv.org/abs/2512.18311.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1321–1330. PMLR, 2017. URL https://proceedings.mlr.press/v70/guo17a.html.

Shibo Hao, Tianyang Liu, Zhen Wang, and Zhiting Hu. ToolkenGPT: Augmenting frozen language models with massive tools via tool embeddings. In *Advances in Neural Information Processing Systems*, volume 36, 2023. doi: 10.52202/075280-1988. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/8fd1a81c882cd45f64958da6284f4a3f-Abstract-Conference.html.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *The Tenth International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Lanxiao Huang, Darshan Nere, Tyler Cody, Peter A. Beling, and Ming Jin. Failure-aware penetration testing via typed failure tokens: From calibrated prediction to structured recovery. In *Conference on Language Modeling*, 2026.

Alon Jacovi and Yoav Goldberg. Towards faithfully interpretable NLP systems: How should we define and evaluate faithfulness? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 4198–4205. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.acl-main.386. URL https://aclanthology.org/2020.acl-main.386/.

Zhengbao Jiang, Jun Araki, Haibo Ding, and Graham Neubig. How can we know when language models know? On the calibration of language models for question answering. *Transactions of the Association for Computational Linguistics*, 9:962–977, 2021. doi: 10.1162/tacl_a_00407. URL https://aclanthology.org/2021.tacl-1.57/.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. SWE-bench: Can language models resolve real-world GitHub issues? In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=VTF8yNQM66.

Manas Joglekar, Jeremy Chen, Gabriel Wu, Jason Yosinski, Jasmine Wang, Boaz Barak, and Amelia Glaese. Training LLMs for honesty via confessions. *arXiv preprint arXiv:2512.08093*, 2025. doi: 10.48550/arXiv.2512.08093. URL https://arxiv.org/abs/2512.08093.

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, Scott Johnston, Sheer El-Showk, Andy Jones, Nelson Elhage, Tristan Hume, Anna Chen, Yuntao Bai, Sam Bowman, Stanislav Fort, Deep Ganguli, Danny Hernandez, Josh Jacobson, Jackson Kernion, Shauna Kravec, Liane Lovitt, Kamal Ndousse, Catherine Olsson, Sam Ringer, Dario Amodei, Tom Brown, Jack Clark, Nicholas Joseph, Ben Mann, Sam McCandlish, Chris Olah, and Jared Kaplan. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*, 2022. doi: 10.48550/arXiv.2207.05221. URL https://arxiv.org/abs/2207.05221.

Sanyam Kapoor, Nate Gruver, Manley Roberts, Katherine Collins, Arka Pal, Umang Bhatt, Adrian Weller, Samuel Dooley, Micah Goldblum, and Andrew Gordon Wilson. Large language models must be taught to know what they don't know. In *Advances in Neural Information Processing Systems*, volume 37, 2024. doi: 10.52202/079017-2729. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/9c20f16b05f5e5e70fa07e2a4364b80e-Abstract-Conference.html.

Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=VD-AYtP0dve.

Kilian Lieret and Carlos E. Jimenez. mini-SWE-agent. Software, version 2.4.3, 2026. URL https://github.com/SWE-agent/mini-swe-agent.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/aca97732e30bcf1303bc22ac3924fd16-Abstract-Conference.html.

Stephanie Lin, Jacob Hilton, and Owain Evans. Teaching models to express their uncertainty in words. *Transactions on Machine Learning Research*, 2022. URL https://openreview.net/forum?id=8s8K2UZGTZ.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by ChatGPT really correct? Rigorous evaluation of large language models for code generation. In *Advances in Neural Information Processing Systems*, volume 36, 2023. doi: 10.52202/075280-0943. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/43e9d647ccd3e4b7b5baab53f0368686-Abstract.html.

David Lopez-Paz, Léon Bottou, Bernhard Schölkopf, and Vladimir Vapnik. Unifying distillation and privileged information. In *The Fourth International Conference on Learning Representations*, 2016. URL https://arxiv.org/abs/1511.03643.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *The Seventh International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Priyanka Mary Mammen, Emil Joswin, and Srujananjali Medicherla. Do agents know when they succeed? Calibrating agent confidence from internal representations. *arXiv preprint arXiv:2609.09448*, 2026. doi: 10.48550/arXiv.2609.09448. URL https://arxiv.org/abs/2609.09448.

METR. Early work on monitorability evaluations. METR research blog, January 2026. URL https://metr.org/blog/2026-01-19-early-work-on-monitorability-evaluations/.

Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, D. Sculley, Sebastian Nowozin, Joshua V. Dillon, Balaji Lakshminarayanan, and Jasper Snoek. Can you trust your model's uncertainty? Evaluating predictive uncertainty under dataset shift. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/8558cb408c1d76621371888657d2eb1d-Abstract.html.

Jiayi Pan, Xingyao Wang, Graham Neubig, Navdeep Jaitly, Heng Ji, Alane Suhr, and Yizhe Zhang. Training software engineering agents and verifiers with SWE-Gym. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 47717–47737. PMLR, 2025. URL https://proceedings.mlr.press/v267/pan25g.html.

Matt Pocock. skills: Skills for real engineers. GitHub repository; accessed June 29, 2026. URL https://github.com/mattpocock/skills.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, 2023. doi: 10.52202/075280-0377. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html.

KaShun Shum, Binyuan Hui, Jiawei Chen, Lei Zhang, X. W., Jiaxi Yang, Yuzhen Huang, Junyang Lin, and Junxian He. SWE-RM: Execution-free feedback for software engineering agents. *arXiv preprint arXiv:2512.21919*, 2025. doi: 10.48550/arXiv.2512.21919. URL https://arxiv.org/abs/2512.21919.

Ningzhi Tang, Chaoran Chen, Gelei Xu, Yiyu Shi, Yu Huang, Collin McMillan, Tao Dong, and Toby Jia-Jun Li. How coding agents fail their users: A large-scale analysis of developer-agent misalignment in 20,574 real-world sessions. *arXiv preprint arXiv:2605.29442*, 2026. doi: 10.48550/arXiv.2605.29442. URL https://arxiv.org/abs/2605.29442.

Katherine Tian, Eric Mitchell, Allan Zhou, Archit Sharma, Rafael Rafailov, Huaxiu Yao, Chelsea Finn, and Christopher Manning. Just ask for calibration: Strategies for eliciting calibrated confidence scores from language models fine-tuned with human feedback. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 5433–5442. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.emnlp-main.330. URL https://aclanthology.org/2023.emnlp-main.330/.

Miles Turpin, Julian Michael, Ethan Perez, and Samuel R. Bowman. Language models don't always say what they think: Unfaithful explanations in chain-of-thought prompting. In *Advances in Neural Information Processing Systems*, volume 36, 2023. doi: 10.52202/075280-3275. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/ed3fea9033a80fea1376299fa7863f4a-Abstract.html.

Dennis Ulmer, Martin Gubri, Hwaran Lee, Sangdoo Yun, and Seong Joon Oh. Calibrating large language models using their generations only. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15440–15459. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.824. URL https://aclanthology.org/2024.acl-long.824/.

You Wang, Michael Pradel, and Zhongxin Liu. Are "solved issues" in SWE-bench really solved correctly? An empirical study. In *Proceedings of the 2026 IEEE/ACM 48th International Conference on Software Engineering*. Association for Computing Machinery, 2026. doi: 10.1145/3744916.3764576. URL https://doi.org/10.1145/3744916.3764576.

Tianyang Xu, Shujin Wu, Shizhe Diao, Xiaoze Liu, Xingyao Wang, Yangyi Chen, and Jing Gao. SaySelf: Teaching LLMs to express confidence with self-reflective rationales. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 5985–5998. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.emnlp-main.343. URL https://aclanthology.org/2024.emnlp-main.343/.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. doi: 10.48550/arXiv.2505.09388. URL https://arxiv.org/abs/2505.09388.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. SWE-agent: Agent-computer interfaces enable automated software engineering. In *Advances in Neural Information Processing Systems*, volume 37, 2024. doi: 10.52202/079017-1601. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/5a7c947568c1b1328ccc5230172e1e7c-Abstract-Conference.html.

Barry Zhang, Keith Lazuka, and Mahesh Murag. Equipping agents for the real world with Agent Skills. Anthropic engineering blog, October 2025. URL https://www.anthropic.com/engineering/equipping-agents-for-the-real-world-with-agent-skills.

Jiaxin Zhang, Caiming Xiong, and Chien-Sheng Wu. Agentic confidence calibration. *arXiv preprint arXiv:2601.15778*, 2026. doi: 10.48550/arXiv.2601.15778. URL https://arxiv.org/abs/2601.15778.

A 72-TASK EVALUATION: SETUP AND DETAILED RESULTS

This evaluation compares generated completion declarations on a held-out set of 72 tasks excluded from both SFT and verdict training. Figure 3 and Table 5 include the SFT baseline, a hard-label-only continuation (Hard-J), and Soft-J. Table 5 also reports process fidelity, task success, and probability metrics. Table 12 adds the independently calibrated Hard-J comparator. Appendix E.2 compares all scorers on each fixed trajectory source.

Task selection. We sampled 72 SWE-bench Verified issues across repositories (seed 0) from 247 issues that were not previously used in development-set evaluations or adapted-model checkpoint selection. The selected 72 issues are disjoint at the issue level from all three SFT data sources (GLM demonstrations, SWE-Gym trajectories, and base-policy examples) and from the verdict-training and verdict-development sets. They are not used for development tuning or checkpoint selection.

SFT, Hard-J and Soft-J rollout configuration. We compare the SFT baseline, Hard-J, and Soft-J (training details in Appendix D.1). Each agent independently generates three rollouts per issue, yielding 216 total episodes per model. Rollouts run on mini-SWE-agent using the source-editing interface described in Appendix C, with sampling temperature 0.7 and an 80-step interaction budget.

Table 5: Results on new rollouts across the 72-task held-out evaluation set, including the generated-verdict comparison in Figure 3. Each agent generates its own execution trajectories. Counts are reported for generated verdicts, task success, and report coverage. The SFT/Soft-J probability populations are detailed in Table 10, and their field-level fidelity statistics appear in Appendix C. Bold marks the best point estimate per performance row: lower FC, gap, ECE, and Brier; higher task success, report accuracy, and AUC. Verdict/outcome rates and report coverage are descriptive.

Measure	SFT baseline	Hard-J	Soft-J
Generated $\mathrm{FC}^{\mathrm{gen}}_{\mathrm{green}}$	25/30 (0.833)	**7/25 (0.280)**	11/28 (0.393)
Positive-verdict rate, green reports	62/69 (0.899)	19/53 (0.358)	31/60 (0.517)
Outcome rate, green reports	39/69 (0.565)	28/53 (0.528)	32/60 (0.533)
Absolute verdict–outcome gap, green	23/69 (0.333)	9/53 (0.170)	**1/60 (0.017)**
Task success, all episodes	**54/216 (0.250)**	53/216 (0.245)	52/216 (0.241)
Report coverage, all episodes	118/216 (0.546)	115/216 (0.532)	114/216 (0.528)
`Etest` report accuracy	**0.788**	0.780	0.772
`Erepro` report accuracy	0.678	0.720	**0.763**
`Egreen` report accuracy	**0.746**	0.740	0.737
`Eregr` report accuracy	0.712	0.730	**0.746**
Verdict ECE	0.203	0.273	**0.111**
Verdict Brier score	0.272	0.267	**0.199**
Verdict green AUC	0.576	0.662	**0.700**

Hard-label comparison. Hard-J uses outcome-only verdict supervision ($t = y$; Appendix D.1). Independent calibration reduces Hard-J's ECE from 0.273 to 0.164 and Brier score from 0.267 to 0.228. When each scorer evaluates its own policy's trajectories, Soft-J is better than this calibrated baseline on all five probability-based metrics: higher overall AUC (0.778 versus 0.753) and green AUC (0.700 versus 0.662), lower ECE (0.111 versus 0.164) and Brier score (0.199 versus 0.228), and a lower thresholded false-positive rate, $\mathrm{FC}^{p_J>0.5}_{\mathrm{green}}$ (0.19 versus 0.31; Table 12). This comparison combines scorer and trajectory-population differences. Cross-scoring retains Soft-J's ECE and Brier advantages for each source, but Hard-J has higher Green AUC on two of the three sources (Appendix E.2). For generated verdicts, Soft-J yields a smaller aggregate verdict–outcome gap (0.017 versus 0.170). Hard-J has a lower generated false-positive rate (0.280 versus 0.393), with its positive-verdict rate below the outcome rate among green reports (19/53 versus 28/53).

Report extraction. The extractor finds the final assistant turn containing a `VERDICT` marker, parses the emitted E and J tokens, and compares them with the full-episode parser flags. This yields the reports in Table 5 (one baseline and six Soft-J episodes contain redundant intermediate markers).

In cases with missing individual fields, missing E tokens are mapped to negative and missing J to `noresolve`. Separately, continuous verdict-probability evaluation requires an unambiguous single marker at the final turn with valid logit positions. This strict filter retains the full baseline report population but excludes one Soft-J report from an unresolved green rollout (detailed sample accountings in Table 10).

Regression-field reporting accuracy is 74.6% for Soft-J and 71.2% for the SFT baseline (compared to 83.5% for the baseline on the 104-task analysis set). Test, green, and regression report accuracies differ between the models by at most 3.5 percentage points. Task success is 24.1% for Soft-J and 25.0% for the baseline (52 vs. 54 successful episodes out of 216 rollouts).

B Failure Diagnosis: Definitions and Counts

The preliminary diagnosis evaluates recorded execution trajectories across all 500 SWE-bench Verified issues. The two GLM conditions cover the full benchmark with three independent rollouts per issue (1,500 episodes each); the Qwen baseline evaluates the benchmark with one rollout per issue (500 episodes). `GLM-5.2-thinking-high` (GLM-5-Team, 2026) uses temperature 1.0 and top-p 0.95, and `Qwen/Qwen3-32B` runs locally at temperature 0.7 and top-p 0.9; both run in mini-SWE-agent 2.4.3 (Lieret & Jimenez, 2026). All runs use the same completion instruction: "On its own line, output EXACTLY `RESOLVED: yes` or `RESOLVED: no`, then one line of evidence: the test command you actually ran THIS session and its real result. Judge only by tests you ran." The skills condition adds instructions adapted from the `diagnosing-bugs` and `tdd` skills of Pocock (2026), written in the Agent Skills format (Zhang et al., 2025), trimmed for non-interactive execution and embedded verbatim in the agent prompt.

Motivating failure patterns. A false claim is a detected positive completion declaration on a benchmark-unresolved rollout. The two agents exhibit different verification histories within these claims (Figure 4): 79.5–88.3% for GLM follow a passing post-edit check, whereas 76.5% for Qwen involve no recognized check. Although positive-claim detection rates and rollout counts differ, GLM also checks more frequently across all episodes (Table 6). The verdict-training experiments instead use the structured-report metrics of §5.

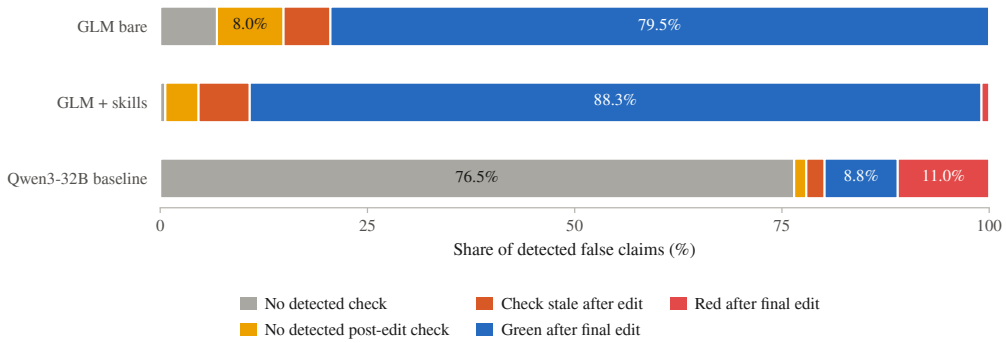


Figure 4: Verification histories within detected false claims (each bar sums to 100%). Green/red denotes a passing/failing result for the last recognized check after the last source edit. All episodes are benchmark-unresolved; claim-detection coverage and repetitions differ by model. Table 6 gives counts; the sensitivity analysis below restricts to explicit hidden-test failures.

Claim detection protocol. The completion detector parses agent assistant turns by prioritizing explicit declarations (`RESOLVED: yes` followed by `RESOLVED: no`), falling back to unambiguous completion phrases outside markdown code blocks when explicit markers are omitted. Of the detected positive claims across conditions (1,307 in GLM bare, 1,248 in GLM skills, and 152 in Qwen), explicit declarations constitute 1,205, 1,038, and 4 cases, respectively, with prose fallbacks accounting for the remainder (102, 210, and 148). GLM usually follows the structured format, whereas unadapted Qwen usually ends with a prose summary, so nearly all Qwen claims come from the prose fallback.

Table 6: Trajectory counts underlying Figure 4. Process categories partition all detected false claims. All-episode verification statistics apply the same execution parser across the complete rollout corpus.

Count	GLM bare	GLM + skills	Qwen baseline
All scored episodes	1500	1500	500
Detected positive claims	1307	1248	152
False claims	336	324	136
Any detected check (all episodes)	1301	1428	84
No detected check (false claims)	23	2	104
No detected post-edit check (false claims)	27	13	2
Check stale after edit (false claims)	19	20	3
Green after final edit (false claims)	267	286	12
Red after final edit (false claims)	0	3	15

Execution flags and verification categorization. The execution parser reads bash commands, test outputs, and return codes from each trajectory. The parser tracks project test runners and standalone reproduction scripts, distinguishing source-code edits from test-script authoring and shell redirections. Command execution status is labeled *green* for return code 0 and *red* for non-zero exit codes, with textual regex matching serving as fallback when exit statuses are unrecorded. Verification states are categorized relative to the first (c_1) and final (c_L) recognized source edits into five mutually exclusive process groups: (1) *no recognized check*: no test or reproduction command executed; (2) *no detected post-edit check*: verification ran prior to editing but not afterward; (3) *check stale after edit*: verification executed after earlier edits but not following the final edit c_L; (4) *green after final edit*: the last recognized check after the final edit c_L was successful; and (5) *red after final edit*: the terminal check after c_L produced a non-zero exit status. Pre-edit checks require a recognized test prior to c_1; pre-edit reproduction requires a failing check before c_1. Table 6 reports the full partition of detected false claims across these states.

Sensitivity to scoring failures. Because benchmark-unresolved rollouts include non-submissions and container execution faults, we repeat the analysis on rollouts with recorded test failures (`FAIL_TO_PASS` or `PASS_TO_PASS`). This filter leaves 329 GLM bare, 319 GLM skills, and 84 Qwen episodes. On this subset, final green checks still dominate for GLM (80.2% bare, 89.0% skills), and claims without a recognized check still dominate for Qwen (77.4%). The contrast between the two agents is unchanged by this restriction.

Prompting more verification. Testing skills increase both pre-edit and post-edit checking, while task success and false claims per episode change little; their confidence intervals include no change (Figure 5, Table 7). The false-claim fraction among detected declarations also remains essentially unchanged. Agent-selected checks can miss issue requirements, leaving a specification gap even when those checks pass.

Table 7: Absolute rates (%) and percentage-point changes underlying Figure 5. Bootstrap intervals account for within-issue clustering across 10,000 resamples. Only the final row conditions on detected positive claims.

Measure	Bare	Skills	Change [95% CI]
Check before first edit	27.1	49.1	+22.0 [+19.3, +24.8]
Red check before first edit	23.7	43.3	+19.7 [+16.9, +22.5]
Check after final edit	75.5	85.7	+10.3 [+7.7, +12.8]
Task success	72.5	73.9	+1.4 [-0.5, +3.4]
False claim / all episodes	22.4	21.6	-0.8 [-2.7, +1.1]
False claim / detected claims	25.7	26.0	+0.3 [-1.9, +2.4]

Statistical uncertainty estimation. To evaluate the effect of skills augmentation (Table 7), we apply cluster bootstrap resampling at the issue level with replacement (10,000 draws, seed 20260916),

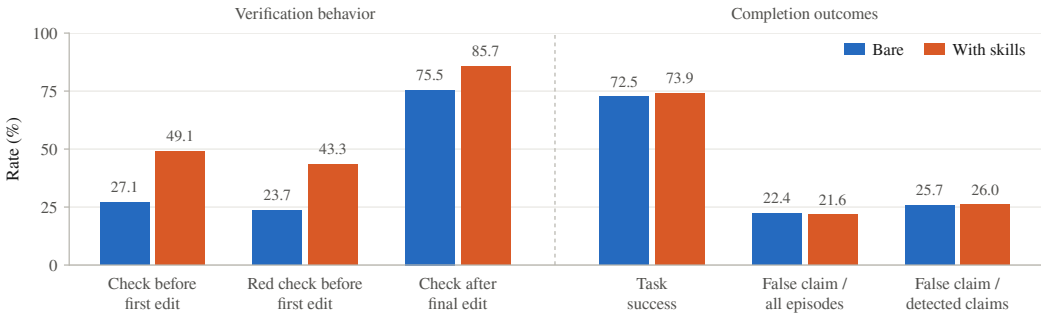


Figure 5: Bare versus skills-augmented GLM: paired bars show absolute rates for verification behavior (left) and completion outcomes (right) on the same issues. Only the final category conditions on detected positive claims. Table 7 gives absolute rates, percentage-point differences, and paired issue-cluster bootstrap 95% confidence intervals.

keeping all rollouts of each issue paired across conditions. For each bootstrap iteration, we recompute absolute rates and percentage-point differences. Confidence intervals are the 2.5th and 97.5th percentiles of the bootstrap distribution; resampling issues accounts for repeated rollouts of the same issue.

C Report Format, Parsers, and Metrics

Token interface. The structured reporting vocabulary consists of opening and closing `VERDICT` tags, two dedicated binary tokens for each of the four E fields, and two binary outcome tokens for J. With a reserved verification gate and five trajectory action markers, the interface adds 18 tokens in total. Reports follow a fixed order: test execution, bug reproduction, post-edit passing check, regression testing, and resolution verdict. For example:

```
<VERDICT> <Etest:y> <Erepro:n> <Egreen:y>
<Eregr:y> <J:noresolve> </VERDICT>
```

Because J follows E, the verdict conditions on the reported evidence, and a negative verdict can follow positive evidence fields, as in this example.

Reference parser. A deterministic execution parser produces the reference labels E^* from bash logs, command return codes, and file diffs. Recognized verification commands include standard test harnesses (e.g., `pytest`, `tox`, `unittest`) and Python scripts with testing or reproduction semantics. Source-code modifications exclude edits to temporary or reproduction scripts and record the first (c_1) and final (c_L) modification steps. Command outcomes follow exit codes (0 for success, non-zero for failure), with regex matching on the output as a fallback. The reference parser defines four binary flags:

- $E^*_{\text{test}} = 1$: At least one recognized test runner or reproduction command was executed.
- $E^*_{\text{repro}} = 1$: A recognized reproduction script was executed prior to the initial source edit c_1 (or at any point if no code edits occurred).
- $E^*_{\text{green}} = 1$: At least one recognized verification command returned success following the final source edit c_L (or at any point if no code edits occurred).
- $E^*_{\text{regr}} = 1$: At least one verification command was executed after the first edit c_1.

The diagnosis in Appendix B uses the result of the last check; all E/J training and fidelity analyses use these definitions.

Fidelity and missing reports. We evaluate report fidelity by comparing generated flags $\widehat{E}_k$ against parser-derived references E^*_k. From the counts $\mathrm{TP}, \mathrm{FP}, \mathrm{FN}, \mathrm{TN}$, we compute accuracy

as $(\mathrm{TP}+\mathrm{TN})/N$, over-claim rate as $\mathrm{FP}/(\mathrm{TP}+\mathrm{FP})$, and under-claim rate as $\mathrm{FN}/(\mathrm{TP}+\mathrm{FN})$. On sparse fields accuracy mostly reflects the majority class, so we also report the two directional rates. Episodes without a report reduce coverage and are not scored as negative reports. Fallback extraction rules for malformed reports are specified in Appendix A.

Table 8: Field-level reporting fidelity across task splits. The 104-task analysis evaluates 322 reports from 624 episodes (92 resolved). Fractions for the 72-task evaluation report exact conditional counts ($\mathrm{FP}/(\mathrm{TP}+\mathrm{FP})$ and $\mathrm{FN}/(\mathrm{TP}+\mathrm{FN})$); reproduction over-claim rates rest on only 15 and 12 positive reports.

Population	Field	Accuracy	Over-claim	Under-claim
104 tasks (322 reports)	Test	0.857	0.06	0.13
	Reproduction	0.702	0.87	0.88
	Green	0.742	0.11	0.36
	Regression	0.835	0.11	0.13
72 tasks: SFT baseline (118)	Test	0.788	11/85	14/88
	Reproduction	0.678	10/15	28/33
	Green	0.746	7/53	23/69
	Regression	0.712	20/85	14/79
72 tasks: Soft-J (114)	Test	0.772	15/81	11/77
	Reproduction	0.763	12/12	15/15
	Green	0.737	6/42	24/60
	Regression	0.746	17/77	12/72

Continuous metrics. To evaluate continuous verdict probabilities p_J, we report Brier score ($N^{-1}\sum_r (p_r - y_r)^2$) and Expected Calibration Error (ECE) for the binary resolution event $y = 1$ (Guo et al., 2017). ECE partitions the predicted probability interval $[0, 1]$ into ten equal-width bins and computes the weighted average difference $|\overline{p}_b - \overline{y}_b|$. Discriminative performance is measured via ROC-AUC, crediting tied predictions with 0.5. Within-issue ranking accuracy evaluates all discordant pairs (one resolved, one unresolved rollout of the same issue), computing the proportion of pairs where the resolved rollout receives a higher probability (ties excluded). Green AUC measures discrimination among rollouts with $E^*_{\mathrm{green}} = 1$. $\mathrm{FC}^{p_J>0.5}_{\mathrm{green}}$ and $\mathrm{FC}^{\mathrm{gen}}_{\mathrm{green}}$ measure false positives among unresolved green rollouts, using thresholded probabilities ($p_J > 0.5$) and generated tokens ($\widehat{J}$), respectively.

D Supervised Fine-Tuning and Verification Behavior

Data and optimization. The supervised fine-tuning (SFT) corpus comprises 13,070 curated trajectory turns: 8,329 from GLM demonstrations, 2,943 from SWE-Gym (Pan et al., 2025), and 1,798 from base-policy rollouts. Prior to filtering, the GLM demonstration pool contains 8,361 turns spanning 941 trajectories collected under the skills-augmented condition. Both the 104-task analysis set and the 72-task final evaluation set are disjoint at the issue level from every component of the SFT corpus: GLM demonstrations, SWE-Gym trajectories, and base-policy rollouts. Training targets cover assistant dialogue turns, tool invocation actions, and structured completion reports.

Source-edit supervision retains the model's native editing format for most examples. Trajectory action labels include `run-test`, `inspect`, `submit`, and `edit`. The GLM source has no separate reproduction action label; the other sources contribute 222 examples of script-based bug reproduction.

Model and training hyperparameters. All agents use `Qwen/Qwen3-32B` (Yang et al., 2025) with rank-16 LoRA (Hu et al., 2022) ($\alpha = 32$, dropout 0, without bias tuning) targeting attention projections (W_q, W_k, W_v, W_o) and feed-forward projections ($W_{\mathrm{gate}}, W_{\mathrm{up}}, W_{\mathrm{down}}$), totaling 134.2M trainable parameters. Base model weights are loaded in `bfloat16` with gradient checkpointing enabled. We use AdamW (Loshchilov & Hutter, 2019) with $\beta = (0.9, 0.999)$, $\epsilon = 10^{-8}$, weight

decay 0.01, gradient clipping at 1.0, microbatch size 1, and seed 0. Table 9 lists the hyperparameters that vary across conditions.

Table 9: Training hyperparameters. LR denotes the initial learning rate scale; steps and warm-up measure optimizer updates. Effective batch size indicates cumulative examples per gradient step. All continuation variants initialize independently from the 800-step SFT baseline checkpoint.

Model / training variant	LR	Steps	Eff. batch	Max. tokens	Warm-up
SFT baseline	5×10^{-4}	800	16	4,096	20
Hard-J / Soft-J	2×10^{-5}	300	8	8,192	10
Soft-only / difficulty-only / no difficulty	2×10^{-5}	300	8	8,192	10
Natural language / J first	2×10^{-5}	300	8	8,192	10
Trajectory reweighting	2×10^{-5}	400	16	4,096	15
Positive E-token incentive	2×10^{-5}	300	8	8,192	10

Initial SFT initializes the 18 added embedding rows from text descriptions of each token and updates them with row-masked gradients. Adapter and embedding parameters use learning rate 5×10^{-4} with linear warm-up and cosine decay to 10% of the initial rate. Continuation variants train only the LoRA adapter, with token embeddings and base weights frozen and the learning rate decayed to zero by a cosine schedule. Truncation keeps the first 800 prompt tokens and the trajectory tail, so supervised targets stay in context. A 4% trajectory-grouped holdout selects the 800-step checkpoint as the SFT baseline.

Task provenance. The 104-task analysis set and 16 initial development issues form a 120-issue source pool for verdict learning. The 16 initial development issues were also excluded from GLM demonstration sourcing. We use the 104-task set to characterize baseline verification behavior and find that the agent's own checks pass in 33.0% of episodes (Table 1). Across the same episodes, 46.0% time out. Rollouts on these issues supply the verdict-learning corpus and the diagnostics in Appendix E.3.

Reproduction behavior and reporting. Auditing GLM demonstrations indicates that explicit pre-edit reproduction commands occur in only 13.6% of training turns, with only 54 of the 333 verdict-training reports possessing a positive strict reproduction label. The baseline agent is similarly sparse: on the 104-task analysis set, it runs pre-edit reproduction in only 17.0% of report-bearing episodes. Manual inspection of 47 reproduction over-claims reveals that 45% involve reproduction scripts executed post-modification or via custom bash scripts not captured by standard test runners, while 55% lack execution support. Broadening the parser definition to accept reproduction commands executed at any trajectory step increases parser-detected occurrences from 17.0% to 48.0%, but introduces 119 false-negative under-claims and lowers overall reporting accuracy from 70.2% to 53.0%. We therefore keep the stricter pre-edit definition.

Targeted reproduction recognition on fixed contexts. We test whether targeted supervision improves reproduction reporting with two continuations from the 300-step Soft-J checkpoint: (1) upweighting positive reproduction examples and temporal hard negatives (reproduction commands executed after editing) for 200 steps; and (2) incorporating GLM trajectory supervision into this mixture for 400 steps. Both variants apply binary cross-entropy only at the `Erepro` position, against strict parser labels (learning rate 10^{-5}, effective batch size 8, context length 8,192).

On the 90 fixed development contexts (21 reproduction-positive, 69 negative), thresholding the reproduction-token probability at 0.5 yields positive reproduction recall of 23.8% (5/21) under reweighting and 52.4% (11/21) under GLM augmentation. Overall accuracy is 78.9% (71/90) for both variants, with false-positive rates of 4.3% and 13.0%, respectively.

D.1 Verdict Targets and Training

The verdict-learning corpus is constructed from rollouts of the baseline agent across the 104-task analysis set and the 16 development issues. Splitting this corpus at the repository issue level (strati-

fied by issue resolution status, seed 0) yields 333 training contexts across 89 issues and 90 development contexts across 22 issues, each requiring a valid `VERDICT` terminal block. Requiring complete four-field reports leaves 322 training and 87 development contexts (the well-formed subset).

Difficulty-conditioned target formulation. Let $S_{i,-r}$ denote the set of submitted rollouts for issue i excluding rollout r, and let π represent the global empirical success rate across training rollouts. With pseudo-observation weight $\alpha = 2$, the leave-one-out task difficulty baseline is given by:

$$\widetilde{p}_{i,-r} = \frac{\sum_{s \in S_{i,-r}} y_s + \alpha\pi}{|S_{i,-r}| + \alpha}, \tag{3}$$

$$a_{i,-r} = \mathrm{logit}(\widetilde{p}_{i,-r}) - b_{-f(i)}^{\top} \overline{E}_{i,-r}^{*}, \tag{4}$$

$$q_r = \sigma(a_{i,-r} + b_{-f(i)}^{\top} E_r^{*}), \tag{5}$$

where probabilities are bounded within $[0.001, 0.999]$ and $\overline{E}_{i,-r}^{*}$ denotes the mean evidence vector across peer rollouts of issue i. The evidence weight vector b is fitted via regularized fixed-effect logistic regression using five-fold cross-fitting: the coefficient vector $b_{-f(i)}$ applied to issue i is estimated on all training folds excluding the fold containing issue i. Optimization runs for 3,000 gradient descent steps (learning rate 0.1, L2 regularization penalty 1.0 on evidence weights and issue intercepts). In the shared notation $t_r = \lambda y_r + (1-\lambda) q_r$, $\lambda \in [0, 1]$. Soft-J uses $\lambda = 0.3$, giving $t_r = 0.3 y_r + 0.7 q_r$; Hard-J uses $\lambda = 1$, giving $t_r = y_r$.

In both verdict continuations, the loss is computed only on the two-token restricted softmax at the J position ($\mathcal{L}_J = -t_r \log p_{J,r} - (1 - t_r) \log(1 - p_{J,r})$), with adapter weights updated over 300 steps (Table 9). We compare checkpoints at 100, 200, and 300 steps on the fixed development contexts and select the 300-step model.

Post-hoc comparators. As a post-hoc calibration baseline, we fit a logistic regression model on the 333 training contexts using the frozen J margin logit, four parser-derived E^* features, and an intercept (5,000 gradient steps, learning rate 0.05, L2 penalty 0.01).

For *Hard-J + independent calibration*, we fit a post-hoc calibrator on a separate calibration set and apply it to Hard-J's verdict probabilities. Neither the fixed development contexts nor the 72-task evaluation set participates in this fit.

Ablation variants. Target ablations use the same update budget: (1) *Hard-only* (Hard-J, $t_r = y_r$) and *Soft-only* ($t_r = q_r$); (2) *Difficulty-only* ($t_r = 0.3 y_r + 0.7 \widetilde{p}_{i,-r}$), which retains the same leave-one-out difficulty estimate and outcome mixture while removing the evidence adjustment (equivalently, setting $b = 0$ in Eq. 1); (3) *No-difficulty* ($t_r = 0.3 y_r + 0.7 q_{\mathrm{pooled},r}$), where q_{pooled} replaces issue intercepts with a single shared intercept c:

$$q_{\mathrm{pooled},r} = \sigma(c + w^{\top} E_r^{*}), \tag{6}$$

fitted via five-fold cross-fitting by issue across training reports (4,000 steps, step size 0.1, L2 penalty 0.01 on w). Interface ablations change the report format: *J-first* reverses the token order (J before E), and *Natural language* replaces the tokens with prose statements (e.g., "ran tests", "reproduced the bug before editing", "tests passed on the final code") ending with "Will the issue be resolved (yes/no)?".

Prompted confidence baselines. Zero-shot prompting baselines replace the completion block on the frozen SFT baseline across the 87 well-formed development contexts: (1) **Binary prompt:** "Question: will the issue be resolved (yes/no)?", reading restricted softmax probabilities over `yes`/`no` logits; and (2) **Verbalized prompt:** "Confidence that the issue is resolved (0-100):", greedily decoding up to five tokens to extract and normalize the numeric percentage value. Neither baseline involves additional training.

E Additional Results and Diagnostics

E.1 Probability Metrics on Fixed Contexts and New Rollouts

Table 10 summarizes the sample populations and evaluation subsets used across the calibration, ablation, and failure monitoring analyses. The fixed development set covers 22 repository issues across 90 total contexts (87 with complete, well-formed reports). The failed-green and failed-no-green columns count unresolved rollouts with and without a parser-verified passing check after the last edit (E^*_{green}); within-issue ranking uses all non-tied discordant pairs of rollouts from the same issue. Discrete report-generation statistics appear separately in Table 5, including the Hard-J/Soft-J comparison in Figure 3.

Table 10: Sample counts for verdict-probability evaluation. Full fixed contexts are used for the main development results and target ablations; the well-formed subset is used for prompted baselines and interface variants. New-rollout rows count trajectories generated by each policy.

Population	Contexts	Resolved	Unresolved	Failed-green	Failed-no-green
Full fixed contexts	90	37	53	29	24
Well-formed subset	87	37	50	28	22
New rollouts: SFT baseline	118	54	64	30	34
New rollouts: Soft-J	113	49	64	27	37

Table 11: Extended fixed-context probability metrics (resolution rates: full set 0.411, subset 0.425). Ranking uses non-tied discordant pairs. Bold marks column-wise best values within each block, including ties: Brier/FC ↓, AUC/ranking ↑; mean p_J is descriptive. Complements Tables 2 and 4; independent calibration is in Table 2.

Method	Brier	Mean p_J	No-green AUC	$\text{FC}^{p_J>0.5}_{\text{no-green}}$	Within-issue ranking (%)
SFT baseline	0.323	0.605	0.553	0.17	78
Post-hoc	0.251	0.418	0.505	**0.00**	75
Soft-J	**0.205**	0.373	0.775	0.12	**90**
Hard only	0.211	0.283	**0.856**	0.04	86
Hard + post-hoc	0.219	0.251	**0.856**	0.04	87
Soft only	0.213	0.388	0.773	0.08	86
Difficulty only	0.217	0.395	0.761	0.08	82
No difficulty	0.223	0.404	0.748	**0.00**	83
SFT baseline (subset)	0.321	0.608	0.553	0.18	77
Soft-J (subset)	**0.208**	0.376	0.765	0.14	90
Natural language	0.225	0.373	**0.902**	**0.09**	**92**
J first	0.217	0.395	0.778	**0.09**	80

On the well-formed subset, the SFT baseline has AUC 0.591, ECE 0.286, Green AUC 0.511, and $\text{FC}^{p_J>0.5}_{\text{green}} = 0.89$; Soft-J has AUC 0.729, ECE 0.076, Green AUC 0.688, and $\text{FC}^{p_J>0.5}_{\text{green}} = 0.14$. Prompted baselines on the same subset reach Green AUC 0.554 ($P(\text{True})$) and 0.518 (verbalized) and assign $p_J > 0.5$ to every failed green run ($\text{FC}^{p_J>0.5}_{\text{green}} = 1.00$). Their within-issue ranking is high (86% and 100% of 27 non-tied pairs), but their ECE exceeds 0.50 (Table 2).

Reliability estimation protocol. Figure 2 uses five equal-mass bins per model. Pointwise 95% confidence bands are estimated via 2,000 cluster bootstrap resamples at the repository issue level (seed 0) with original bin boundaries held fixed. Expected Calibration Error (ECE) is reported using ten uniform equal-width bins, matching the conventions in Tables 2 and 4.

On the 72-task rollouts, Soft-J is better calibrated than SFT (Table 12). The SFT baseline's mean p_J is 0.594 against a success rate of 0.458; Soft-J's is 0.388 against 0.434. Without passing checks, Soft-J raises No-green AUC from 0.654 to 0.848 and lowers $\text{FC}^{p_J>0.5}_{\text{no-green}}$ from 0.26 to 0.05; the generated false-claim rate without green checks falls from 0.24 to 0.11.

Table 12: Continuous probability metrics on newly generated rollouts across the 72-task held-out evaluation set. Metrics use each model's independently generated trajectory distribution. SFT/Soft-J sample denominators appear in Table 10. Bold marks the best reported value in each column.

Model	AUC	Brier	ECE	Green AUC	$\mathrm{FC}_{\mathrm{green}}^{p_J>0.5}$
SFT baseline	0.672	0.272	0.203	0.576	0.83
Hard-J	0.753	0.267	0.273	0.662	0.31
Hard-J + independent calibration	0.753	0.228	0.164	0.662	0.31
Soft-J	**0.778**	**0.199**	**0.111**	**0.700**	**0.19**

Additional rollouts on the 104-task set. We also ran Soft-J twice on each task in the 104-task set (35/208 episodes resolved, 95 valid reports). Soft-J has a generated $\mathrm{FC}_{\mathrm{green}}^{\mathrm{gen}}$ of 0.24 (8/33), predicting a 0.43 positive verdict rate against an empirical success rate of 0.38 in the green subgroup. Partitioning across issue splits, the 17 issues belonging to the verdict-development set contribute 34 episodes and 6 failed-green reports ($\mathrm{FC}_{\mathrm{green}}^{\mathrm{gen}} = 2/6$, with positive verdict and success rates matching at 0.45). The other 87 issues yield 174 episodes and an $\mathrm{FC}_{\mathrm{green}}^{\mathrm{gen}}$ of 6/27; 78 of them have verdict-training examples, and nine have no usable verdict-training or development contexts. Because this set overlaps verdict training, the 72-task set, which is excluded from SFT, verdict training, and verdict development, remains the primary evaluation.

E.2 Cross-Scoring Held-Out Trajectories

We score the fixed completion contexts from each of the SFT, Soft-J, and Hard-J policies with all four scorers. The trajectory source determines the history, generated E prefix, and outcome; these remain unchanged across scorers within a source. Table 13 reports the full matrices, including pooled results. The SFT and Soft-J sources contain 118 and 113 probability contexts, respectively (Table 10). These metrics condition on usable completion contexts rather than all rollout episodes.

Soft-J has lower ECE and Brier score than independently calibrated Hard-J for all three sources. The Brier difference is small on Hard-J trajectories (0.226 versus 0.228). Hard-J has higher Green AUC on SFT and Soft-J trajectories, whereas Soft-J leads on Hard-J trajectories. At the 0.5 threshold, uncalibrated Hard-J has fewer failed-green false positives on SFT trajectories, while Soft-J has fewer on the other two sources. Relative to independently calibrated Hard-J, Soft-J ties on SFT trajectories and has fewer false positives on both other sources and in the pooled results. Calibration favors Soft-J on every source, whereas discrimination and thresholded decisions depend on the source.

E.3 Evidence Dependence and Coverage

We compare resolved and unresolved rollouts within each of the 27 issues in the 104-task set that have both. Among pairs that differ in whether a check passes, the passing check belongs to the resolved rollout in 95.1% of cases (117 vs. 6 pairs). Similar within-issue associations hold across individual verification actions: 90 vs. 10 pairs for general test execution, 94 vs. 10 for post-edit checking. When only one of two rollouts of an issue runs or passes a check, it is usually the resolved one.

A pooled analysis of generated binary completion declarations combines 322 reports from the 104-task set with 79 reports from the development set (401 total). Across this pooled sample, the overall AUC is 0.625; when stratifying by identical parser-derived evidence vectors E^*, the weighted average AUC is 0.555. Grouping reports by the count of verified evidence fields (100, 190, and 41 reports) yields positive verdict rates of 0.29, 0.85, and 0.93, compared to ground-truth resolution rates of 0.28, 0.51, and 0.46. Positive verdicts rise with the number of verified fields much faster than resolution does.

To test whether low coverage explains failures after passing checks, we measure differential line coverage across 182 green rollouts from the 104-task set (110 unresolved, 71 resolved; 1 omitted due to coverage tracing faults). Hidden benchmark test suites cover a mean of 74.7% of edited lines in unresolved rollouts compared to 79.5% in resolved rollouts, yielding an AUC of 0.541 (complete

Table 13: Full cross-scoring results on trajectories from the 72-task held-out evaluation. Rows fix the trajectory source and columns vary the scorer; Hard-J+cal uses independent calibration. Bold marks the best point estimate per row for directional metrics. The outcome rates for the SFT and Soft-J sources are 0.458 and 0.434; the actual outcome rates for the Hard-J and pooled sources are 0.46 and 0.45. Thresholded probability decisions differ from sampled, generated verdicts.

Trajectory source	SFT scorer	Hard-J	Hard-J+cal	Soft-J
ECE ↓				
SFT	0.203	0.258	0.171	**0.118**
Soft-J	0.220	0.254	0.158	**0.111**
Hard-J	0.207	0.273	0.164	**0.142**
Pooled	0.210	0.261	0.164	**0.123**
Brier score ↓				
SFT	0.272	0.249	0.237	**0.212**
Soft-J	0.260	0.258	0.226	**0.199**
Hard-J	0.256	0.267	0.228	**0.226**
Pooled	0.263	0.258	0.230	**0.212**
Green AUC ↑				
SFT	0.576	**0.688**	**0.688**	0.650
Soft-J	0.624	**0.719**	**0.719**	0.700
Hard-J	0.667	0.662	0.662	**0.681**
Pooled	0.631	**0.690**	**0.690**	0.678
Failed-green $\mathrm{FC}_{\mathrm{green}}^{p_J>0.5}$ ↓				
SFT	0.83	**0.17**	0.20	0.20
Soft-J	0.78	0.26	0.29	**0.19**
Hard-J	0.72	0.31	0.31	**0.24**
Pooled	0.78	0.24	0.26	**0.21**
Mean p_J *(descriptive; no preferred direction)*				
SFT	0.594	0.304	0.390	0.374
Soft-J	0.572	0.327	0.397	0.388
Hard-J	0.615	0.322	0.404	0.416
Pooled	0.594	0.318	0.397	0.392

line coverage occurs in 62/110 and 46/71 rollouts, respectively). The same analysis on GLM gives an AUC near 0.50. Line coverage therefore separates resolved from unresolved green rollouts only weakly.

Table 14 reports counterfactual interventions that toggle one reported evidence field while holding the execution history fixed. For negative-to-positive flips of the test, green, and regression fields, Soft-J reduces the increase in verdict confidence relative to SFT.

Table 14: Counterfactual report interventions across 87 contexts with complete reports. Each entry reports the mean change in predicted success probability (Δp_J) when toggling a single evidence token while holding the execution trajectory fixed. For negative-to-positive flips of the test, green, and regression fields, Soft-J's increase is 63–69% smaller than the baseline's.

	Negative → Positive			Positive → Negative		
		SFT baseline			SFT baseline	
Field	n	Δp_J	Soft-J Δp_J	n	Δp_J	Soft-J Δp_J
Test	23	+0.399	+0.123	64	+0.009	+0.054
Reproduction	70	+0.088	+0.055	17	+0.055	+0.004
Green	42	+0.348	+0.130	45	−0.084	−0.016
Regression	23	+0.441	+0.157	64	+0.032	+0.147

E.4 Target Sensitivity to the Reproduction Field

We refit the reliability target without `Erepro`, keeping folds, regularization, and difficulty baselines fixed. Across 333 training reports from 89 issues, the three-field and four-field targets have Pearson correlation $r = 0.99$. Their outcome AUCs are 0.919 and 0.912, respectively (Table 15).

Table 15: Soft-J reliability target with and without the reproduction field. Dropping `Erepro` leaves the target's outcome AUC nearly unchanged.

Reliability target	Reports	AUC against outcomes
All four evidence fields	333	0.912
Without `Erepro`	333	0.919

E.5 Incentivizing Positive Reports

We compare two ways of encouraging verification on the 22 verdict-development issues (6 rollouts per issue, 132 episodes per model): (1) SFT that upweights successful trajectories with parser-verified checks and truthful report labels, and (2) positive token rewards applied directly to the evidence fields E. Both start from the SFT baseline (hyperparameters in Table 9). Table 16 reports verification behavior and task success for both incentives and the SFT baseline.

Table 16: Behavioral and outcome rates under the two verification incentives, over all scored episodes. Report fidelity for each scheme is given in the text.

Behavioral metric	SFT baseline	Trajectory reweighting	Positive-token incentive
Passing checks	0.417	0.439	0.523
Any recognized test	0.591	0.614	0.697
Task success	0.250	0.273	0.303
Scored episodes	132	132	132

Under direct token incentives, all 86 emitted reports claim positive verification across all four fields ($E = (y, y, y, y)$) with positive completion verdicts. Consequently, `Erepro` report accuracy collapses to 23.3%, and the false-positive verdict rate among unresolved cases reaches 100%. In contrast, trajectory reweighting increases passing checks while preserving report over-claim rates at $\leq$8% on audited fields. Rewarding report tokens directly raises claimed verification far more than actual verification.

E.6 RETROSPECTIVE MONITORING

We use $1 - p_J$ as a failure-risk score. Following the fixed-false-positive-budget protocol of METR (2026), we measure failure detection at nominal false-positive rates (FPR) of 10% and 20% on successful runs. An episode is flagged when its p_J falls below the α-quantile of p_J on successful completions; catch rate is computed over all usable completion contexts.

Table 17: Failure detection performance under fixed false-positive budgets. Bootstrap confidence intervals are derived from 2,000 issue-cluster resamples, accounting for trajectory correlation within issues.

Population	Score source	Catch@10%	Catch@20%	95% Bootstrap CI at 20%
Fixed contexts (90)	SFT baseline	0.28	0.34	[0.21, 0.56]
	Soft-J	**0.32**	**0.51**	[0.22, 0.82]
72 tasks	SFT baseline	0.38	0.45	[0.30, 0.64]
	Soft-J	**0.45**	**0.59**	[0.39, 0.79]
Well-formed subset (87)	Prompted $P(\mathrm{True})$	0.18	0.26	—
	Prompted verbalized	0.24	0.24	—

Soft-J catches more failures than SFT at both budgets on fixed contexts and new rollouts (Table 17), although the intervals at 20% overlap. Prompted confidence catches 24–26% of failures at 20% on the well-formed subset.

F CASE STUDIES OF EVIDENCE REPORTING AND OUTCOME JUDGMENT

We examine four SFT baseline trajectories from the 104-task analysis set, covering the four combinations of process reporting fidelity ($\widehat{E}$ vs. E^*) and outcome verdict correctness ($\widehat{J}$ vs. y). Timelines summarize bash actions; E vectors follow the order `(Etest, Erepro, Egreen, Eregr)`. Hidden evaluation outcomes are computed offline.

Case A: Faithful Process Report, Correct Outcome Verdict ($\widehat{E}$ Faithful, $\widehat{J}$ Correct) `django__django-12143`, sample 3.

s1–s2: Inspect `admin/options.py`; modify regex pattern construction to properly escape URL prefix.
s3–s9: Attempt initial test execution; resolve environment dependencies, missing settings, and import paths.
s10–s12: Execute targeted regression suite via `admin_inlines.tests`: `Ran 59 tests; OK (skipped=9)`; export git diff and submit.
Report: E = `(y,n,y,y)`, J = `resolve`.
Reference: E^* = `(y,n,y,y)`, Ground Truth: `resolved`.

Analysis: Source edits precede testing, so `Erepro:n` is correct. The passing test run supports the other three evidence fields, and the hidden evaluation also passes.

Case B: Faithful Process Report, Overconfident False-Positive Verdict ($\widehat{E}$ Faithful, $\widehat{J}$ False Positive) `scikit-learn__scikit-learn-13124`, sample 4.

s1–s17: Inspect repository; write and execute a standalone reproduction script `repro.py` prior to editing.
s18–s20: Edit documentation and default parameter handling for `StratifiedKFold`; rerun reproduction script and `test_split.py`: `61 passed`.
s21–s22: Attempt broader test execution; runner fails due to non-existent target paths.
s23–s25: Run `model_selection/tests/`: `172 passed`; export patch and submit.
Report: E = `(y,y,y,y)`, J = `resolve`.
Reference: E^* = `(y,y,y,y)`, Ground Truth: `unresolved`.

Analysis: All four evidence flags match the execution record. Despite the passing checks, the patch fails the hidden test `test_shuffle_stratifiedkfold`, making the positive verdict incorrect.

Case C: Reproduction Reporting Over-Claim, Correct Outcome Verdict ($\widehat{E}$ Over-claimed, $\widehat{J}$ Correct) `django__django-16255`, sample 2.

s1–s2: Inspect `sitemaps/__init__.py`; broaden exception handling block from `TypeError` to `(TypeError, ValueError)`.
s3–s14: Attempt test execution; configure environment paths and test runner settings.
s15–s17: Run project test suite `sitemaps_tests`: `Ran 60 tests; OK`; export patch and submit.
Report: E = `(y,y,y,y)`, J = `resolve`.
Reference: E^* = `(y,n,y,y)`, Ground Truth: `resolved`.

Analysis: No reproduction command precedes the initial edit, making `Erepro:y` an over-claim; the post-edit checks are reported accurately. The hidden test `test_callable_sitemod_no_items` passes, so the positive verdict is correct.

Case D: Unsupported Evidence Hallucination, False-Positive Verdict ($\widehat{E}$ Hallucinated, $\widehat{J}$ False Positive) `django__django-15380`, sample 5.

s1–s2: Inspect `migrations/autodetector.py`; modify model naming logic.
s3–s4: Review source diff, export patch, and submit. No test runner or reproduction command was executed at any point in the trajectory.
Report: E = `(y,y,y,y)`, J = `resolve`.
Reference: E^* = `(n,n,n,n)`, Ground Truth: `unresolved`.

Analysis: The agent reports all four checks although it ran none, and the patch fails hidden evaluation (`test_rename_field_with_renamed_model`). Both the evidence report and the positive verdict are incorrect.